महाराष्ट्र राज्य तंत्रशिक्षण मंडळ, मुंबई
(स्वायत्त) (ISO 21001:2018) (ISO/IEC 27001:2013)

अभियांत्रिकी आणि तंत्रज्ञान पदविका

शिक्षण पुस्तिका
(Learning Material)

OPERATING SYSTEM
(315319)

K Scheme

ऑपरेटिंग सिस्टिम

संगणक अभियांत्रिकी गट
(के स्कीम)

मराठी–इंग्रजी (द्विभाषिक) माध्यम
(अभियांत्रिकी व तंत्रज्ञानातील पाचवे सत्र पदविका)

शिक्षण पुस्तिका

(Learning Material)

ऑपरेटिंग सिस्टिम

Operating System

(315319)

संगणक अभियांत्रिकी गट

(के-स्कीम)

K-Scheme

मराठी-इंग्रजी (द्विभाषिक) माध्यम

(अभियांत्रिकी व तंत्रज्ञानातील पाचवे सत्र पदविका)

महाराष्ट्र राज्य तंत्रशिक्षण मंडळ, मुंबई
(स्वायत्त) (ISO 21001:2018)(ISO/IEC 27001:2013)

ऑपरेटिंग सिस्टिम
Operating System
(315319)

मार्गदर्शक

प्रा. विजय नामदेवराव कुकरे

विभागप्रमुख, संगणक अभियांत्रिकी

प्रकल्प समन्वयक

प्रा. विजय नामदेवराव कुकरे

विभागप्रमुख, संगणक अभियांत्रिकी

संकलक

प्रा. विजय नामदेवराव कुकरे

विभागप्रमुख, संगणक अभियांत्रिकी

महाराष्ट्र राज्य तंत्र शिक्षण मंडळ.

(स्वायत्त) (ISO: २१००१:२०१८) (ISO/IEC: २७००१-२०१३)

शासकीय तंत्रनिकेतन इमारत, चौथा मजला, ४९, खेरवाडी, बांद्रा (पूर्व), मुंबई - ४०० ०५१.

दूरध्वनी क्र.: ०२२-६२५४२१०० / १५३ / १७०

email : director@msbte.com web site : www.msbte.ac.in

प्रास्ताविक

महाराष्ट्र राज्यातील पदविका स्तरावरील तंत्रशिक्षणामध्ये विद्यार्थ्यांचे रोजगार कौशल्य विकसित करून विद्यार्थ्यांचा सर्वांगीण विकास घडवून आणण्याकरिता महाराष्ट्र राज्य तंत्रशिक्षण मंडळ कटिबद्ध आहे. उद्योगधंद्यातील बदलत्या तंत्रज्ञानाशी संबंधित गरजा लक्षात घेऊन महाराष्ट्र राज्य तंत्र शिक्षण मंडळाकडून पदविका अभ्यासक्रम वेळोवेळी अद्यावत करण्यात येतो. अभियांत्रिकी पदविका अभ्यासक्रम शिकत असताना संकल्पनात्मक ज्ञान, सुसंगत संदर्भ, प्रश्न विचारणे, विश्वसनीय पुरावे, कारणमीमांसा आणि सुस्पष्ट निकष यांचा वापर करून अर्थाची उकल करण्याची, विश्लेषण व मूल्यमापन करण्याची तसेच तर्काने अनुमान काढण्याची क्षमता म्हणजेच चिकित्सक विचार विद्यार्थ्यांमध्ये अधिक दृढ होतील असा मला विश्वास आहे. जेव्हा विद्यार्थी ज्ञान मिळवण्याच्या माध्यमाशी पूर्णपणे परिचित आणि सोयीस्कर असतात, तेव्हा त्यांच्यासाठी वर्गातील चर्चेत भाग घेणे सोपे होते, संकल्पनात्मक व सैद्धांतिक बाबींचे आकलन परिपूर्ण होते, संज्ञानात्मक क्षमता सुधारते आणि त्यांचा आत्मविश्वास देखील वाढतो. या सर्व गोष्टींचा विचार करून मंडळाकडून शैक्षणिक सामुग्रीची निर्मिती करण्यात आलेली आहे. भारत देश हा खेड्यापाडयातून विकसित झालेला देश असून ग्रामीण भागातील विद्यार्थ्यांना तांत्रिक शिक्षण घेताना भाषेचा अडसर न येता तांत्रिक बाबींचा आशय समजून घेणे शक्य होईल या दृष्टिकोनातून महाराष्ट्र राज्य तंत्र शिक्षण मंडळाने पदविका स्तरावरील तांत्रिक शिक्षणाकरीता विद्यार्थ्यांना मराठी-इंग्रजी द्विभाषिक माध्यमाचा पर्याय उपलब्ध करुन दिलेला आहे.

राष्ट्रीय शैक्षणिक धोरण-२०२० प्रादेशिक भाषेतील शिक्षणास प्रोत्साहन देते, ज्यामुळे विद्यार्थ्यांना तांत्रिक अभ्यासक्रमांसाठी प्रादेशिक भाषेतुन शिक्षणाचे माध्यम निवडता येते. त्या अनुषंगाने प्रादेशिक भाषांमध्ये तांत्रिक सामग्री आणि अभ्यास सामग्रीचा विकास आणि भाषांतर करण्याची आवश्यकता आहे. या धोरणास अनुसरुन मंडळाने भागधारकांसाठी शैक्षणिक वर्ष २०२१-२२ पासून I-Scheme तसेच शैक्षणिक वर्ष २०२३-२४ पासून K-Scheme मध्ये व्दिभाषिक माध्यमाचा पर्याय प्रथम ते तृतीय वर्षाकरिता उपलब्ध करून दिलेला आहे. या पर्यायास अनुसरून मंडळाने मराठी-इंग्रजी द्विभाषिक शैक्षणिक सामग्रीही संबंधीत विद्यार्थी व अधिव्याख्यातांकरिता उपलब्ध करून दिली आहे.

पदविका स्तरावरील तंत्रशिक्षण अधिक दर्जेदार करण्यासाठी महाराष्ट्रातील अनुभवी व तज्ञ अध्यापकांनी व्यवहारिक मराठी भाषा व इंग्रजी भाषेतील तांत्रिक शब्दावली यांचा वापर करून मराठी इंग्रजी भाषेचा सुवर्णमध्य साधण्याचा प्रयत्न केलेला आहे. मंडळाच्या स्तरावर गठीत सुकाणू समितीमार्फत सदर शैक्षणिक सामुग्रीचा दर्जा, तसेच इतर बाबींची तपासणी करण्यात आलेली आहे. त्यामुळे सदर शैक्षणिक सामुग्री अधिक संपन्न झालेली असून,विद्यार्थी त्यांच्या व्यक्तिमत्त्वाचा सुसंवादी आणि सर्वांगीण विकास साधतील. परिणामतः विश्वस्तरीय मनुष्यबळाच्या गरजा पूर्ण करण्यात महाराष्ट्र राज्य अग्रेसर राहील व पर्यायाने राष्ट्रनिर्मिती करीता निश्चितच हातभार लागेल, असा मला विश्वास आहे.

अभियांत्रिकी पदविका अभ्यासक्रमातील विषयांची मराठी-इंग्रजी (द्विभाषिक) शैक्षणिक सामुग्री बनविण्यासाठी अध्यापक व सुकाणू समितीचे सदस्य यांनी दर्शविलेले समर्पण व वचनबद्धता कौतुकास पात्र आहे, या सर्वांचे मी मनःपूर्वक अभिनंदन करतो!

(डॉ. प्रमोद नाईक)
संचालक
म. रा. तंत्र शिक्षण मंडळ, मुंबई

अनुक्रमणिका

युनिट-1
ऑपरेटिंग सिस्टम सेवा आणि घटक
(Operating System services and components)

विषय निष्पत्ती (Course Outcome):

CO1: ऑपरेटिंग सिस्टमच्या सेवा आणि घटकांचे स्पष्टीकरण द्या.

घटक निष्पत्ती (Theory Learning Outcome-TLO):

1. ऑपरेटिंग सिस्टमच्या कार्ये वर्णन करा.
2. ऑपरेटिंग सिस्टमच्या वेगवेगळ्या सेवा स्पष्ट करा.
3. ऑपरेटिंग सिस्टमच्या सिस्टम कॉलचा वापर स्पष्ट करा.
4. ऑपरेटिंग सिस्टमच्या घटकांशी संबंधित ऍक्टिव्हिटीजचे स्पष्टीकरण द्या.

1.1 ऑपरेटिंग सिस्टमची ओळख (Introduction to Operating System)

ऑपरेटिंग सिस्टम कोणत्याही संगणक सॉफ्टवेअरचा एक आवश्यक भाग आहे. ऑपरेटिंग सिस्टमशिवाय आपण संगणक ऑपरेट करू शकत नाही. ऑपरेटिंग सिस्टमचा मुख्य हेतू म्हणजे संगणकाच्या हार्डवेअरशी संवाद साधण्यासाठी, सिस्टमचे हार्डवेअर आणि सॉफ्टवेअर संसाधने मॅनेजमेंट करण्यासाठी, अप्लिकेशन सॉफ्टवेअर सक्षम करणे. ऑपरेटिंग सिस्टम (Operating System) संगणक वापरकर्ता आणि संगणक हार्डवेअर दरम्यान एक इंटरफेस आहे. ऑपरेटिंग सिस्टम एक सॉफ्टवेअर आहे, जे फाइल मॅनेजमेंट, मेमरी मॅनेजमेंट, प्रोसेस मॅनेजमेंट, इनपुट आणि आउटपुट हाताळणी आणि डिस्क ड्राइव्हज आणि प्रिंटर सारख्या पेरिफेरल उपकरणांवर नियंत्रण ठेवणारी सर्व मूलभूत कार्ये करते. काही लोकप्रिय ऑपरेटिंग सिस्टममध्ये लिनक्स ऑपरेटिंग सिस्टम (Linix Operating System), विंडोज ऑपरेटिंग सिस्टम (Windows Operating System), व्हीएमएस (VMS), ओएस/400 (OS/400), एआयएक्स (AIX), झेड/ओएस (Z/OS) इ. समाविष्ट आहे.

1.1.1 ऑपरेटिंग सिस्टमची व्याख्या (Definition of Operating System)

ऑपरेटिंग सिस्टम हा प्रोग्रामचा एक संच आहे जो युझर आणि संगणक हार्डवेअर दरम्यान इंटरफेस म्हणून कार्य करतो आणि सर्व प्रकारच्या प्रोग्राम्सच्या एक्सिक्युशनवर नियंत्रण ठेवतो.

किंवा

ऑपरेटिंग सिस्टम सिस्टम सॉफ्टवेअरचा एक सेट किंवा संग्रह आहे जो संगणकाची संसाधने मॅनेज करण्यासाठी वापरला जातो, जसे प्रोसेसर, मेमरी, फाइल, डिस्क, इनपुट/आउटपुट डिव्हाइस इत्यादी व संगणक आणि यूजर दरम्यान इंटरफेस प्रदान करतात.

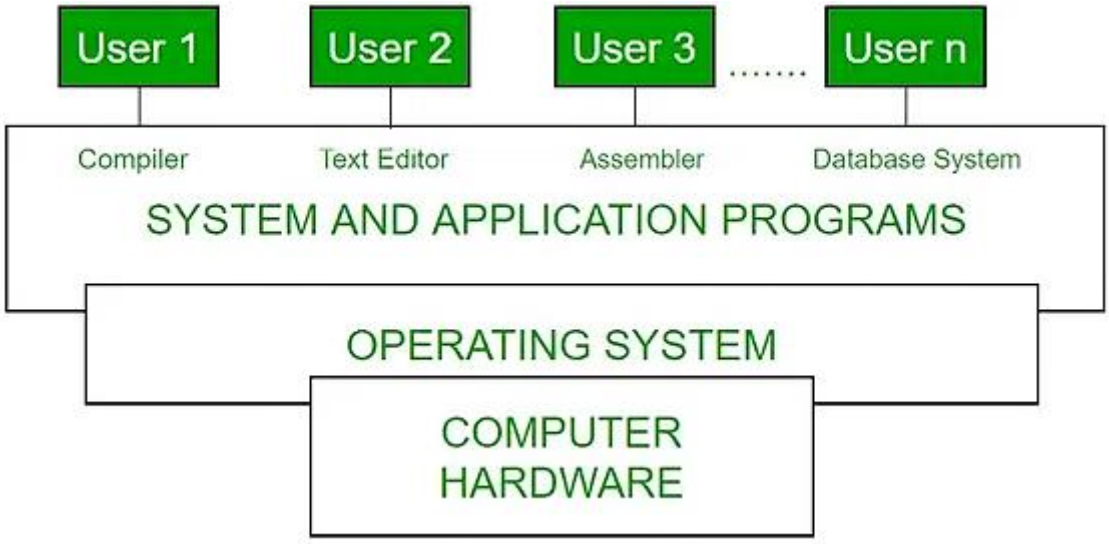


Fig.1.1: ऑपरेटिंग सिस्टिम (Operating System)

1.1.2 ऑपरेटिंग सिस्टमची कार्ये (Functions of an Operating System)

ऑपरेटिंग सिस्टमची काही महत्त्वपूर्ण कार्ये खालीलप्रमाणे आहेत.

1. **मेमरी मॅनेजमेंट (Memory Management)**
   - मेमरी मॅनेजमेंट फंक्शन एकतर वाटप केलेल्या (Allotted) किंवा फ्री (Free) असलेल्या प्रत्येक मेमरी लोकशनच्या स्थितीचा मागोवा ठेवते.
   - हे कॉम्पिटिन्ग प्रोसेसेसला किती मेमरी द्यायची, कोणाला मेमरी मिळेल, केंव्हा मिळेल व किती मिळेल ठरवते.
2. **प्रोसेसर मॅनेजमेंट (Processor Management)**
   - ऑपरेटिंग सिस्टम प्रोसेसरचे कार्य विविध प्रोसेसला वाटप करून मॅनेज करते आणि प्रत्येक प्रोसेसला योग्यरित्या कार्य करण्यासाठी प्रोसेसरकडून पुरेसा वेळ प्राप्त मिळेल याची खात्री करुन प्रोसेसरचे कार्य मॅनेज करते.
   - प्रोसेसचा स्टेटसचा मागोवा ठेवतो आणि जो प्रोग्रॅम हे कार्य करते त्याला प्रोसेस ट्राफिक कंट्रोलर म्हणतात
   - प्रोसेसरला प्रोग्राम प्रोसेस करण्यासाठी सीपीयूचा वेळ अलोकेट करतो.
   - जेव्हा प्रोसेसला CPU ची आवश्यकता नसते तेव्हा प्रोसेसर डी-ॲलोकेट्स करतो.
3. **डिव्हाइस मॅनेजमेंट (Device Management)**
   - सिस्टमशी कनेक्ट केलेल्या सर्व डिव्हाइसचा मागोवा ठेवतो.
   - इनपुट/आउटपुट कंट्रोलर म्हणून ओळखल्या जाणाऱ्या प्रत्येक डिव्हाइससाठी योग्य प्रोग्राम नियुक्त करतो.
   - कोणत्या प्रोसेसला विशिष्ट डिव्हाइसवर किती काळ एक्सेस मिळावा हे ठरवतो.
   - डिव्हाइस प्रभावीपणे आणि कार्यक्षमतेने वाटप करते. जेव्हा प्रोसेसला यापुढे डिव्हाइसची आवश्यकता नसते तेव्हा डिव्हाइसला डी-ॲलोकेट करते.
   - हे या डिव्हाइसकडून रिक्वेस्ट प्राप्त करते, एक विशिष्ट टास्क करते आणि रिक्वेस्टिंग प्रोसेससशी परत कम्यूनिकेशन करते.
4. **फाईल मॅनेजमेंट (File Management)**

   फाइल मॅनेजमेंट ही ऑपरेटिंग सिस्टमची सर्वात व्हिझिबल सेवा आहे. संगणक माहिती (Infromation) बऱ्याच वेगवेगळ्या भौतिक स्वरूपात संचयित करू शकतात; मॅग्नेटिक टेप, डिस्क आणि ड्रम हे सर्वात सामान्य प्रकार आहेत. फाइल मॅनेजमेंटच्या संदर्भात खालील ऍक्टिव्हिटीसाठी ऑपरेटिंग सिस्टम जबाबदार आहे.
   - फाइल निर्मिती आणि हटविणे. (File Creation and Deletion)
   - डिरेक्टरी निर्मिती आणि हटविणे. (Directory Creation and Deletion)
   - फायली आणि डिरेक्टरी हाताळण्यासाठी कमांडचे समर्थन देणे.
   - सेकण्डरी स्टोरेजवर फाईल्स मॅपिंग करणे. (Mapping files onto secondary storage)
   - स्थिर (नॉनव्होटाईल) स्टोरेज मीडियावर फाइल बॅकअप करणे.
5. **सुरक्षा (Security)**

   ऑपरेटिंग सिस्टम युझर डेटा आणि तत्सम इतर तंत्रांचे संरक्षण करण्यासाठी पासवर्डचे संरक्षण वापरते. हे प्रोग्राम आणि वापरकर्त्याच्या डेटामध्ये अनधिकृत प्रवेश प्रतिबंधित करते. ऑपरेटिंग सिस्टम विविध तंत्रे प्रदान करते, जी युझरचा डेटाची अखंडता आणि गोपनीयतेचे आश्वासन देते. खालील सुरक्षा उपायांचा वापर युझरचा डेटा संरक्षित करण्यासाठी केला जातो:
   - लॉगिनद्वारे अनधिकृत प्रवेशाविरूद्ध संरक्षण.
   - फायरवॉल (Firewall) सक्रिय ठेवून घुसखोरीपासून संरक्षण.
   - दुर्भावनायुक्त (Malicious Access) प्रवेशापासून सिस्टम मेमरीचे संरक्षण करणे.
   - सिस्टम असुरक्षा (System Vulnerabilities) संबंधित संदेश प्रदर्शित करणे.

6. **सिस्टम कामगिरीवर नियंत्रण ठेवणे (Control over system performance)**
   ऑपरेटिंग सिस्टम सिस्टम परफॉर्मन्स नियंत्रित आणि ऑप्टिमाइझ (Optimize) करण्यात महत्त्वपूर्ण भूमिका बजावते. ते हार्डवेअर आणि सॉफ्टवेअर दरम्यान मध्यस्थ म्हणून कार्य करतात, हे सुनिश्चित करते की संगणकीय संसाधनांचा (Resources) कार्यक्षमतेने उपयोग केला जातो. एक मूलभूत पैलू म्हणजे संसाधन वाटप (Resource allocation), जेथे ऑपरेटिंग सिस्टम वेगवेगळ्या प्रक्रियेमध्ये सीपीयू वेळ (CPU Time), मेमरी आणि आय/ओ (I/O) डिव्हाइसचे वाटप करते, निष्पक्ष आणि इष्टतम संसाधन (Fair and Optimal) उपयोग प्रदान करण्याचा प्रयत्न करते. प्रोसेस शेड्यूलिंग (Process Scheduling), क्रिटिकल फंक्शन (Critical Function), सीपीयूची मक्तेदारी करण्यापासून आणि प्रभावी मल्टीटास्किंग (Multitasking) सक्षम करण्यापासून कोणत्याही एकाच टास्कला प्रतिबंधित करताना कोणत्या प्रोसेस किंवा थ्रेड (Thread) रन कराव्या हे ठरविण्यात मदत करते.
7. **एरर डिटेक्टीन्ग एड्स (Error detecting aids)**
   ऑपरेटिंग सिस्टम त्रुटी (Error) शोधण्यासाठी आणि संगणक प्रणाली खराब होण्यापासून टाळण्यासाठी सतत सिस्टमचे परीक्षण करते. वेळोवेळी, ऑपरेटिंग सिस्टम कोणत्याही एक्सटर्नल थ्रेट (External Threat) किंवा मालिसीअस सॉफ्टवेअर (Malicious Software) ऍक्टिव्हिटीसाठी सिस्टमची तपासणी करते आणि कोणत्याही प्रकारच्या नुकसानीसाठी हार्डवेअर देखील तपासते. ही प्रोसेस युजरला कित्येक सतर्कता दर्शविते जेणेकरून सिस्टमला होणाऱ्या कोणत्याही नुकसानीविरूद्ध योग्य कारवाई केली जाऊ शकते.
8. **इतर सॉफ्टवेअर आणि युझर्स मधील समन्वय (Coordination between other software and users)**
   ऑपरेटिंग सिस्टम संगणक प्रणालीच्या विविध युझर्सला इंटरप्रिटर , कंपाईलर, असेंबलर्स आणि इतर सॉफ्टवेअरचे समन्वय आणि नियुक्त करतात. सोप्या शब्दांत, ऑपरेटिंग सिस्टमचा आपल्या संगणकाचा ट्रॅफिक कॉप म्हणून विचार करा. भिन्न सॉफ्टवेअर प्रोग्राम आपल्या संगणकाची संसाधने कशी शेअर करू शकतात हे निर्देशित करते आणि व्यवस्थापित करते. हे सुनिश्चित करते की जेव्हा आपल्याला एखादा प्रोग्राम वापरायचा असेल तेव्हा ते इतरांना क्रॅश न करता किंवा समस्या न आणता सहजतेने चालते.
9. **जॉब अकाउंटिंग (Job accounting)**
   ऑपरेटिंग सिस्टम विविध टास्क आणि युझर्स द्वारे वापरल्या जाणाऱ्या वेळ आणि संसाधनांचा मागोवा ठेवते, ही माहिती विशिष्ट युझर्ससाठी किंवा युझर्सच्या ग्रुप साठी संसाधन वापराचा मागोवा घेण्यासाठी वापरली जाऊ शकते. मल्टीटास्किंग (Multitasking) ओएसमध्ये जिथे एका पेक्षा अधिक प्रोग्राम्स एकाच वेळी चालतात, तिथे ओएस निर्धारित करते की कोणत्या ऑप्लिकेशन्सना कोणत्या क्रमाने चालवावे आणि प्रत्येक ऑप्लिकेशनला वेळ कसा वाटला पाहिजे.

1.2 विविध प्रकारचे ऑपरेटिंग सिस्टम (Different Types of Operating Systems)

ऑपरेटिंग सिस्टम अगदी पहिल्या संगणक निर्मितीपासून आहेत आणि ते वेळेसह विकसित होत आहेत. आता आम्ही ऑपरेटिंग सिस्टमच्या काही महत्त्वाच्या प्रकारच्या चर्चा करू, जे सामान्यत: वापरले जातात.

1.2.1 बॅच ऑपरेटिंग सिस्टम (Batch operating system)

बॅच ऑपरेटिंग सिस्टमचे युझर्स थेट संगणकाशी संवाद साधत नाहीत. प्रत्येक युझर्स पंच कार्ड (Punch Card) सारख्या ऑफ-लाइन डिव्हाइसवर आपला जॉब तयार करतो आणि ते संगणक ऑपरेटरला सबमिट करतो जे Fig. 1.2 मध्ये दर्शविले आहे. जॉब प्रोसेसिंगला गती देण्यासाठी, समान गरजा असलेले जॉब्स एकत्र करून आणि गट म्हणून रन केले जातात. प्रोग्रामर ऑपरेटरला त्यांचे प्रोग्राम रन करण्यासाठी देतात आणि नंतर ऑपरेटर समान आवश्यकतेसह प्रोग्रामची बॅचमध्ये क्रमवारी लावतात.

बॅच सिस्टममधील समस्या (Problems) खालीलप्रमाणे आहेत:

1. युझर्स आणि जॉब्स दरम्यान परस्परसंवादाचा अभाव. (Lack of Interaction)
2. सीपीयू (CPU) बऱ्याचदा निष्क्रिय (Ideal) असतो, कारण मेकॅनिकल I/O डिव्हाइसची गती सीपीयूपेक्षा कमी असते.
3. जॉब्स किंवा युझर्सला इच्छित प्राधान्य (Desired Priority) प्रदान करणे कठीण.

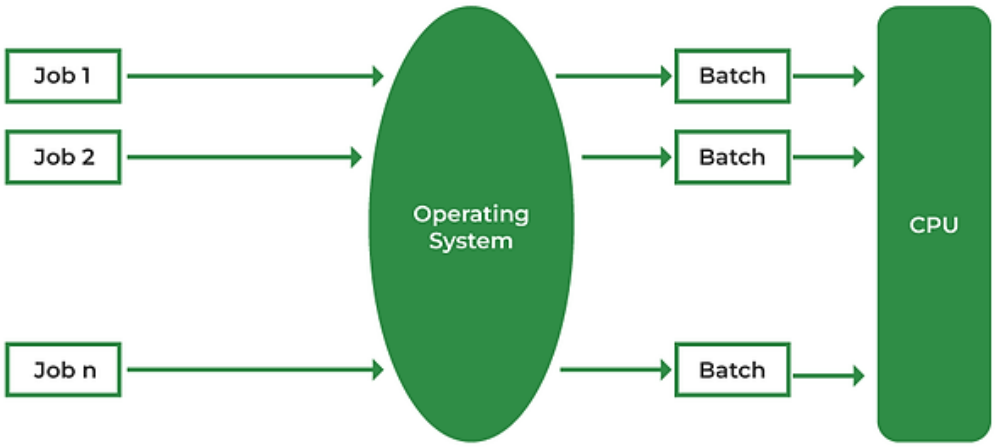


Fig.1.2: बॅच ऑपरेटिंग सिस्टम (Batch Operating System)

बॅच ऑपरेटिंग सिस्टमचे फायदे (Advantages of batch operating systems)

1. युझर्सचा इंटरॅक्शन शिवाय बॅच सिस्टममध्ये रिपीटेड जॉब्स वेगवान रन केले जातात.
2. बॅच सिस्टममध्ये डेटा इनपुट करण्यासाठी विशेष हार्डवेअर आणि सिस्टम सपोर्ट आवश्यकता नाही.
3. निष्क्रिय (Ideal) वेळ बॅच सिस्टम खूपच कमी असते.
4. बॅच सिस्टमच्या प्रोसेसरला क्यूमध्ये (Queue) असतानाही जॉब्सच्या कालावधीची माहिती असते.
5. बॅच सिस्टम एका पेक्षा अधिक युझर्सद्वारे शेअर (Share) केले जाऊ शकतात.
6. बॅच ऑपरेटिंग सिस्टम आपल्याला कार्यक्षमतेने मोठ्या प्रमाणात काम व्यवस्थापित करण्यास सक्षम करते.
7. प्रोसेस पूर्ण केली जाते, जॉब स्पूलमधील पुढील जॉब कोणत्याही युझर्सच्या इंटरॅक्शनशिवाय एक्झिक्युट केला जातो.
8. सीपीयूचा उपयोग सुधारित (Improved) होतो.

बॅच ऑपरेटिंग सिस्टमचे तोटे (Disadvantages of batch Operating systems)

1. प्रोग्राम डीबग (Debug) करणे कठीण असते.
2. जॉब इंफायनाईट (Infinite) लूपमध्ये प्रवेश करू शकते.
3. प्रोटेक्शन (Protection) स्कीमचा कमतरतेमुळे, एका बॅचच्या जॉब प्रलंबित जॉब वर परिणाम करू शकतो.
4. बॅच सिस्टम महाग (Costly)असते.
5. जर कोणताही जॉब अपयशी (Fails) ठरला तर त्याच्या एक्झिक्युशन वेळेचा अंदाज लावणे कठीण असते.

बॅच आधारित ऑपरेटिंग सिस्टमची उदाहरणे (Example): IBM's z/OS जे पेरोल सिस्टम, बँक स्टेटमेन्ट इ. वापरली जाते

1.2.2 मल्टीप्रोग्रामिंग (मल्टी प्रॉग्रॅम्ड) ऑपरेटिंग सिस्टम (Multiprogramming [Multi-programmed] Operating System)

मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टम एकाच प्रोसेसर संगणकावर बरेच प्रोग्राम रन करू शकते. एखाद्या प्रोग्रामने मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टममध्ये इनपुट/आउटपुट ट्रान्सफरची प्रतीक्षा करणे आवश्यक असल्यास, इतर प्रोग्राम सीपीयू (CPU) वापरण्यास तयार असतात. परिणामी, विविध जॉब्स सीपीयूचा वेळ शेअर करू शकतात, तथापि, त्यांच्या जॉब्सचे एक्झिक्युशन एकाच वेळी असल्याचे परिभाषित केलेले नाही. जेव्हा एखादा प्रोग्राम रन केला जात असेल तेव्हा तो "टास्क" (Task), "प्रोसेस"(Process) आणि "जॉब"(Job) म्हणून ओळखला जातो. कॉनकरंट प्रोग्राम (Concurrent program) एक्झीक्युएशन्स सीरियल आणि बॅच प्रोससिंग प्रणालीच्या तुलनेत सिस्टम संसाधनाचा वापर आणि थ्रूपुट (Throughput) सुधारतात. मल्टीप्रोग्रामिंगचे प्राथमिक लक्ष्य संपूर्ण सिस्टमची संसाधने मॅनेज करणे आहे. मल्टीप्रोग्रामिंग सिस्टमचे मुख्य घटक म्हणजे फाइल सिस्टम, कमांड प्रोसेसर, ट्रान्झिएंट एरिया आणि आय/ओ कंट्रोल सिस्टम आहेत. रिसोर्स मॅनेजमेंट रुटीनस ऑपरेटिंग सिस्टम कोर (Core) फंक्शन्सशी जोडल्या जातात.

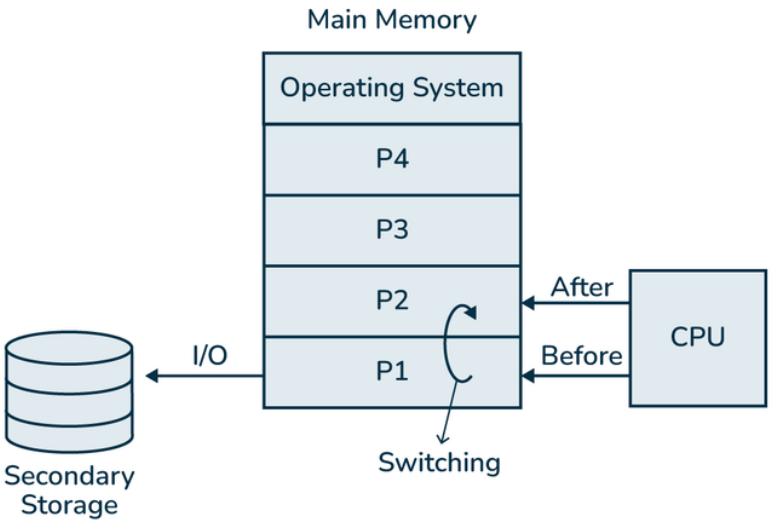


Fig.1.3: मल्टीप्रोग्रामिंग (मल्टी प्रॉग्रॅम्ड) ऑपरेटिंग सिस्टम Multiprogramming (Multi-programmed) Operating System

मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टमचे फायदे: (Advantages of multiprogramming operating system)

1. सीपीयूचा (CPU) वापर जास्त होतो कारण सीपीयू कधीही निष्क्रिय (Ideal) स्थितीत जात नाही.
2. मेमरीचा (Memory) वापर आणि उपयोग पूर्ण कार्यक्षमतेणे करतो.
3. सीपीयूचे थ्रूपुट (Throuput) उच्च असते आणि एका पेक्षा अधिक इंटरॅक्टिव्ह (Interactive) युझर्सच्या टर्मिनलला सपोर्ट करते.

मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टमचे तोटे: (Disadvantages of multiprogramming operating system)

1. सीपीयू शेड्यूलिंग (CPU Scheduling)अनिवार्य आहे कारण बरेच जॉब्स एकाच वेळी सीपीयूवर रन होण्यास तयार असतात.
2. युझर जॉब एक्झिक्युट होतांना जॉबशी इंटरॅक्ट करणे शक्य नसते
3. प्रोग्रामर एक्झिक्युट होत असलेल्या प्रोग्राममध्ये मॉडिफिकेशन करू शकत नाहीत.

मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टमचे प्रकार (Types of Multiprogramming Operating Systems)

मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टमचे दोन प्रमुख प्रकार खाली दिलेले आहेत.

1. मल्टीटास्किंग ऑपरेटिंग सिस्टम (Multitasking Operating System)
2. मल्टीयुझर ऑपरेटिंग सिस्टम (Multiuser Operating System)

1. मल्टीटास्किंग ऑपरेटिंग सिस्टम (Multitasking Operating System)

मल्टीटास्किंग ओएस दोन किंवा अधिक प्रोग्राम्सचे एकाचवेळी ऑपरेशन करण्यास सक्षम करते. ऑपरेटिंग सिस्टम प्रत्येक प्रोग्रामला एका वेळी मेमरीमध्ये किंवा मेमरीमधून बाहेर काढून हे करते. मेमरीमधून स्विच केलेला प्रोग्राम पुन्हा एकदा आवश्यक होईपर्यंत डिस्कवर तात्पुरते जतन केला जातो.

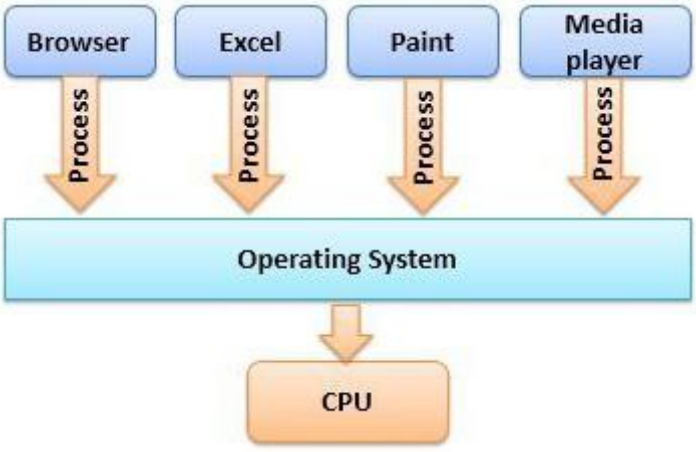


Fig. 1.4: मल्टीटास्किंग ऑपरेटिंग सिस्टम (Multitasking Operating System)

उदाहरण: मायक्रोसॉफ्ट विंडोज (Microsoft Windows), आयबीएम ओएस/ 390 (IBM's OS/390), आणि लिनक्स (Linux).

मल्टीटास्किंग ऑपरेटिंग सिस्टम चे फायदे: (Advantages of Multi-tasking Operating System)

1. मल्टी-टास्किंग ऑपरेटिंग सिस्टम, सिस्टम स्लोव डाउन न करता एकाच वेळी एका पेक्षा अधिक अप्लिकेशन एक्झिक्युट करण्यास सक्षम आहे.
2. प्रत्येक प्रोसेसला विशिष्ट लांबीची सीपीयूचा कालावधी दिली जाते (म्हणजे वेळ शेअर करणे), म्हणूनच सीपीयूचा वापर करण्यासाठी प्रोसेसला जास्त काळ प्रतीक्षा करावी लागत नाही.
3. मल्टीटास्किंग ओएस आय/ओ डिव्हाइस (I/O Deice), रॅम (RAM), हार्ड डिस्क (Hard Disk), सीपीयू (CPU) आणि इतर संगणक संसाधने (Resources) प्रभावीपणे व्यवस्थापित करू शकते.
4. मल्टी-टास्किंग ऑपरेटिंगमध्ये यूजर एकाच वेळी एका पेक्षा अधिक प्रोग्राम्स एक्झिक्युट करू शकतो, जसे की गेम्स (Games), ब्राउझर (Browser), एमएस वर्ड (MS Word) आणि इतर सेवा.

मल्टीटास्किंग ऑपरेटिंग सिस्टम चे तोटे: (Disadvantages of Multi-tasking Operating System)

1. एक प्रोसेसर एकाच वेळी एका पेक्षा अधिक प्रोसेस एक्झिक्युट करीत असताना सीपीयू वर लोड येते आणि सीपीयूची गरम होऊ शकतो.
2. एकाच वेळी एका पेक्षा अधिक प्रोग्राम्स एक्झिक्युट करताना मल्टीटास्किंग ऑपरेटिंग सिस्टममध्ये प्रोसेसर धीमा (Slow) असल्यास संगणक सिस्टम सुद्धा धीमी (Slow) होते.
3. मेन मेमरी (RAM) ला एका पेक्षा अधिक प्रोसेसेस मल्टीटास्किंग दरम्यान स्टोअर कराव्या लागतात ज्यामुळे मुख्य मेमरी ओव्हरलोड होते कारण मेमरीची मर्यादा असू शकते.

2. मल्टीयुझर ऑपरेटिंग सिस्टम (Multiuser Operating System)

मल्टीयुझर ऑपरेटिंग सिस्टममधील विविध टर्मिनलवरील बरेच यूजर्स एक शक्तिशाली मध्यवर्ती संगणक शेअर करू शकतात हे ऑपरेटिंग सिस्टमद्वारे टर्मिनल दरम्यान जलदपणे स्विच करते, त्यापैकी प्रत्येकास मुख्य संगणकावर विशिष्ट प्रमाणात प्रोसेसरचा वेळ वाटप केला जातो. मल्टिझर ऑपरेटिंग सिस्टममध्ये, एका पेक्षा अधिक संख्येने यूजर्स एकाच वेळी संगणकाच्या भिन्न रिसोर्स एक्सेस करू शकतात. मेनफ्रेम संगणक (Mainfram Computer System) प्रणालीशी संलग्न असलेल्या विविध पर्सनल संगणकांचा समावेश असलेल्या नेटवर्कचा वापर करून एक्सेस केला जातो. मल्टियूजर ऑपरेटिंग सिस्टम एका वेळी एकाच मशीनला एक्सेस करण्यासाठी एकाधिक यूज़र्सच्या परवानगी देते. विविध पर्सनल संगणक (Personal Computer) मेनफ्रेम संगणक प्रणालीला माहिती पाठवू आणि प्राप्त करू शकतात. अशाप्रकारे, मेनफ्रेम संगणक सर्व्हर आणि इतर पर्सनल संगणक त्या सर्व्हरसाठी क्लायंट (Client) म्हणून कार्य करतो म्हणून कार्य करतो.

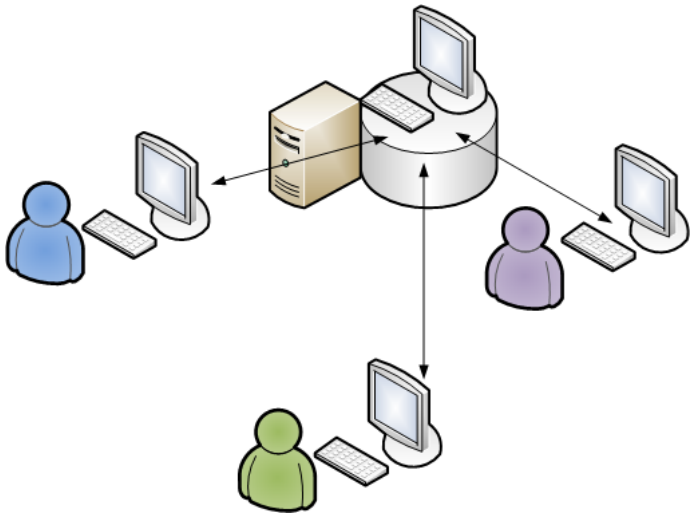


Fig. 1.5: मल्टीयुझर ऑपरेटिंग सिस्टम (Multiuser-Operating System)

उदाहरण: उबंटू लिनक्स (Ubantu Linux), यूनिक्स (Unix), मॅक ओएस एक्स (Mac OS X), मायक्रोसॉफ्ट विंडोज सर्वर (Microsoft Windows Server), नॉवेल (Novell)

मल्टीयुझर ऑपरेटिंग सिस्टमचे फायदे: (Advantages of Multiuser Operating System)

1. वेगवेगळ्या यूजर्सचा डेटा आणि माहिती शेअरिंग करण्यास मदत करते.
2. प्रिंटरसारख्या हार्डवेअर संसाधनांच्या शेअरिंग करण्यात देखील मदत करते.
3. व्यत्यय टाळते जर कोणताही एक-संगणक फेल झाला तर तो त्या नेटवर्कवर उपस्थित असलेल्या इतर संगणकावर परिणाम करत नाही.
4. यूजर्स त्यांचे कार्य इतर यूजर्स सोबत शेअर करू शकतात.
5. मल्टी-यूजर ऑपरेटिंग सिस्टममध्ये डेटा बॅक अप केला जाऊ शकतो.
6. मल्टी-यूजर ऑपरेटिंग सिस्टमच्या सेवा खूप स्थिर आणि पद्धतशीर असतात

मल्टीयुझर ऑपरेटिंग सिस्टमचे तोटे: **(Disadvatages of Multiuser Operating System)**

1. मेनफ्रेम संगणक सेट करण्यासाठी महाग हार्डवेअर आवश्यक आहे.
2. जेव्हा एका पेक्षा अधिक यूजर्स त्याच सिस्टमवर लॉग इन करतात किंवा काम करतात तेव्हा ते सिस्टमची एकूण कामगिरी कमी करते.
3. माहिती लोकांमध्ये शेअर केली जाते म्हणून येथे गोपनीयता एक चिंता बनते.

1.2.3 मल्टीप्रोसेसर सिस्टिम (Multiprocessor system)

ऑपरेटिंग सिस्टममध्ये, एका संगणक सिस्टिम मध्ये एकापेक्षा जास्त सीपीयू (CPU) कार्यक्षमता सुधारण्यासाठी वापरली जाऊ शकते ज्याला मल्टीप्रोसेसर ऑपरेटिंग सिस्टम म्हणतात. एका पेक्षा अधिक सीपीयू एकमेकांशी जोडलेले असतात जेणेकरून वेगवान (Fast) एक्झिक्युशन साठी त्यांच्यात जॉब (Job) विभाजन केले जाऊ शकते. जेव्हा एखादा जॉबचे एक्झिक्युशन संपते, तेव्हा सर्व सीपीयूचे रिझल्ट (Result) एकत्रित केले जातात आणि अंतिम आउटपुट देण्यासाठी संकलित केले जातात. मुख्य मेमरी शेअर करण्यासाठी आवश्यक असलेले जॉब्स (Jobs) आपापसांत इतर सिस्टिमचे संसाधने (Resources) देखील शेअर करू शकतात. एका पेक्षा अधिक सीपीयू एकाच वेळी एका पेक्षा अधिक जॉब्स रन करण्यासाठी साठी देखील वापरले जाऊ शकतात. उदाहरणार्थ, युनिक्स ऑपरेटिंग सिस्टम (UNIX Operating System) सर्वात मोठ्या प्रमाणात वापरल्या जाणाऱ्या मल्टीप्रोसेसिंग सिस्टमपैकी एक आहे. मल्टीप्रोसेसिंग सिस्टमची मूलभूत रचना खाली दिलेल्या Fig.1.6 मध्ये दर्शविली आहे.

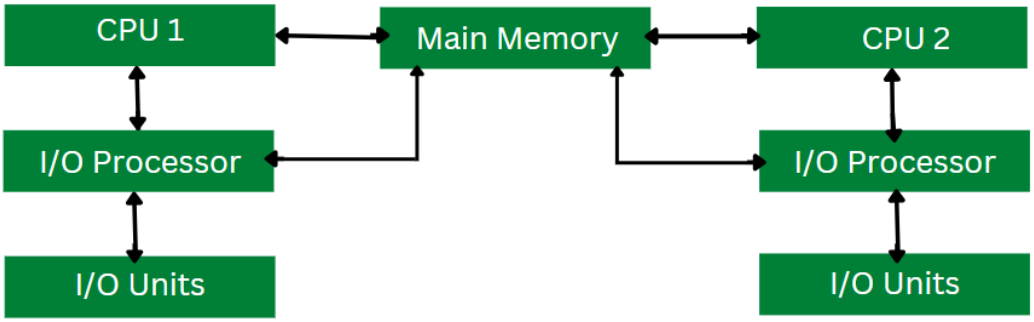


Fig. 1.6: मल्टीप्रोसेसिंग ऑपरेटिंग सिस्टम (Multiprocessing Operating system)

उदाहरण: यूनिक्स (UNIX), लिनक्स (LINUX), आणि सन सोलारिस (Sun Solaris)

मल्टीप्रोसेसर सिस्टमचे फायदे (Advantages of Multiprocessor system)

1. **थ्रूपुट वाढवते (Increase Throughput)**: मल्टीप्रोसेसिंग एकाचवेळी अधिक प्रोसेस एक्झिक्युट करू शकते, ज्यामुळे उच्च थ्रूपुट होते.
2. **सुधारित प्रतिसाद (Improved Responsiveness)**: पॅरलल प्रोसेसिंग (Parallel Processing) मुळे टास्क वेगवान हाताळली जाऊ शकतात, परिणामी अधिक चांगले सिस्टम प्रतिसाद मिळते.
3. **फॉल्ट टॉलरन्स (Fault Tolerance)**: जर एक प्रोसेसर फेल झाला तर, सिस्टम उर्वरित प्रोसेसरसह कार्य करत राहू शकते, ज्यामुळे विश्वसनीयता सुधारते.

4. **चांगली स्केलेबिलिटी (Better Scalability):** अधिक प्रोसेसर जोडले तर वाढीव वर्कलोड हाताळण्यासाठी सिस्टमची क्षमता वाढवू शकते.
5. **एन्हान्सड रिलायबिलिटी (Enhanced Reliability):** एका पेक्षा अधिक प्रोसेसर असल्यास सिस्टम फेल होण्याची शक्यता कमी होते.

मल्टीप्रोसेसर सिस्टमचे तोटे (Disavantages of Multiprocessor system)

1. **वाढलेली जटिलता (Increased Complexity)**: एक प्रोसेसर सिस्टिम पेक्षा एका पेक्षा अधिक प्रोसेसर असलेल्या सिस्टिममध्ये इंटरॅक्शन आणि मॅनेजमेंट करणे अधिक जटिल आहे.
2. **जास्त खर्च (Higher Costs)**: मल्टीप्रोसेसर सिस्टमला सामान्यत: अधिक महाग हार्डवेअर आणि सॉफ्टवेअर आवश्यक असते.
3. **सिंक्रोनाइझेशनची आवश्यकता (Need for Synchronization)**: एका पेक्षा अधिक प्रोसेसरच्या ऍक्टिव्हिटीचे समन्वय साधण्यासाठी आणि कॉन्फ्लिक्ट (Conflict) रोखण्यासाठी विशेष अल्गोरिथम (Algorithm) आवश्यक असते.
4. **डेडलॉकची संभाव्यता (Potential for Deadlocks)**: जर प्रोसेसर एकाच वेळी शेअर्ड संसाधनांचा एक्सेस करण्याचा प्रयत्न करीत असतील तर प्रोग्रेस थांबवून डेडलॉक होऊ शकतात.
5. **असमान वर्कलोड वितरण (Uneven Workload Distribution)**: प्रोसेसरमध्ये वर्कलोड समान रीतीने वितरित केले गेले आहेत हे सुनिश्चित करणे आव्हानात्मक असू शकते.
6. **जास्त मेमरीची आवश्यकता (Increase Memory Requirements)**: मल्टीप्रोसेसर सिस्टमला बऱ्याचदा एका पेक्षा अधिक प्रोसेसर आणि शेअर्ड मेमरीला सपोर्ट देण्यासाठी अधिक मेमरीची आवश्यकता असते.
7. **ऑपरेटिंग सिस्टमची जटिलता (Complexity of Operating System)**: ओएस एका पेक्षा अधिक प्रोसेसर आणि त्यांचे इंटरॅक्शन इफिशिअंटली मॅनेज करण्यासाठी डिझाइन केलेले असणे आवश्यक आहे.

1.2.4 डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टम (Distributed operating System)

डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टम एक प्रकारची ऑपरेटिंग सिस्टम आहे जी हार्डवेअर संसाधने म्यानेज करण्यासाठी आणि एका पेक्षा अधिक रीअल-टाइम ऍप्लिकेशन्स आणि युझर्सला सेवा देण्यासाठी एका पेक्षा अधिक सेंट्रल प्रोसेसर वापरते. डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टमसह, प्रोसेस करावयाचे सर्व जॉब्स एका पेक्षा अधिक प्रोसेसरमध्ये वितरित केल्या जातात. डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टममधील प्रत्येक सिस्टम किंवा नोडची स्वतःची मेमरी तसेच स्वतःचे प्रोसेसर असते. डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टममधील प्रत्येक सिस्टम किंवा नोड विविध कम्यूनिकेशन लाईन्स (जसे की हाय-स्पीड बसेस किंवा टेलिफोन लाइन) द्वारे एकमेकांशी कम्युनिकेट करतात. याला लुजली कपल्ड सिस्टिम (Loosely coupled systems) किंवा डिस्ट्रीब्युटेड सिस्टिम म्हणून संबोधले जाते. डिस्ट्रीब्युटेड सिस्टममधील प्रोसेसरचा आकार आणि फंक्शनमध्ये वेगवेगळे असू शकतात. या प्रोसेसरला साइट्स, नोड्स (Nodes), संगणक इत्यादी म्हणून संबोधले जाते. डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टम Fig 1.7 मध्ये दर्शविली आहे.

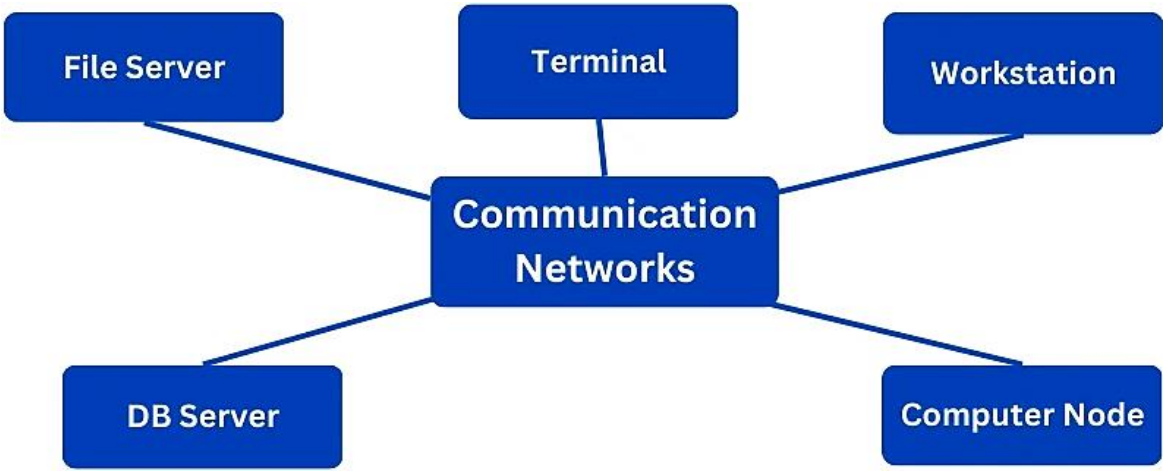


Fig.1.7: डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टम (Distributed operating System)

डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टमचे उदाहरण:

1. विंडोज सर्वर 2003 (Windows server 2003)
2. विंडोज सर्वर 2008 (Windows server 2008)
3. विंडोज सर्वर 2012 (Windows server 2012)
4. उबंटू लिनक्स (Ubuntu Linux)
5. लिनक्स अपाचे सर्वर (Linux Apache Server)

डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टमचे फायदे (Advantages of Distributed Operating System)

1. एका सिस्टिमच्या फेल झाल्या मुळे इतर नेटवर्क कम्यूनिकेशनवर परिणाम होत नाही, कारण सर्व सिस्टम एकमेकांपासून स्वतंत्र आहेत.
2. इलेक्ट्रॉनिक मेल डेटा एक्सचेंजची गती वाढवते.
3. संसाधने (Resources) शेअर केली जात असल्याने कॉम्पुटेशन अत्यंत वेगवान आणि टिकाऊ असते.
4. होस्ट (Host) संगणकावर लोड कमी होते.
5. या सिस्टिम सहजपणे स्केलेबल आहेत कारण नेटवर्कमध्ये बऱ्याच सिस्टम सहजपणे जोडल्या जाऊ शकतात.
6. डेटा प्रोसेसिंगमध्ये विलंब कमी होतो.

डिस्ट्रीब्युटेड ऑपरेटिंग सिस्टमचे तोटे (Disadvantages of Distributed Operating System)

1. मुख्य नेटवर्क फेल झाले कि संपूर्ण कम्यूनिकेशन थांबते
2. डिस्ट्रीब्युटेड सिस्टम स्थापित करण्यासाठी वापरली जाणारी लँग्वेज चांगली परिभाषित केलेली नसते
3. या प्रकारच्या सिस्टम सहज उपलब्ध नाहीत कारण ते खूप महाग असते. केवळ अंडरलाईंग सॉफ्टवेअर अत्यंत कॉम्प्लेक्स असते आणि सध्यातरी ती चांगली समजली नाही

1.2.5 नेटवर्क ऑपरेटिंग सिस्टिम (Network operating System)

नेटवर्क ऑपरेटिंग सिस्टम सर्व्हरवर (Server) चालते आणि सर्व्हरला डेटा, यूजर्स , ग्रुप , सुरक्षा, अप्लिकेशन आणि इतर नेटवर्किंग फंक्शन्स व्यवस्थापित करण्याची क्षमता प्रदान करते. नेटवर्क ऑपरेटिंग सिस्टमचा प्राथमिक हेतू नेटवर्कमधील सामान्यत: लोकल एरिया नेटवर्क (LAN), खाजगी नेटवर्क किंवा इतर नेटवर्कमध्ये, एका पेक्षा अधिक संगणकांमध्ये शेअर्ड फाइल आणि प्रिंटर वापरण्याची अनुमती देणे आहे. नेटवर्क ऑपरेटिंग सिस्टम Fig. 1.8 मध्ये दर्शविली आहे.

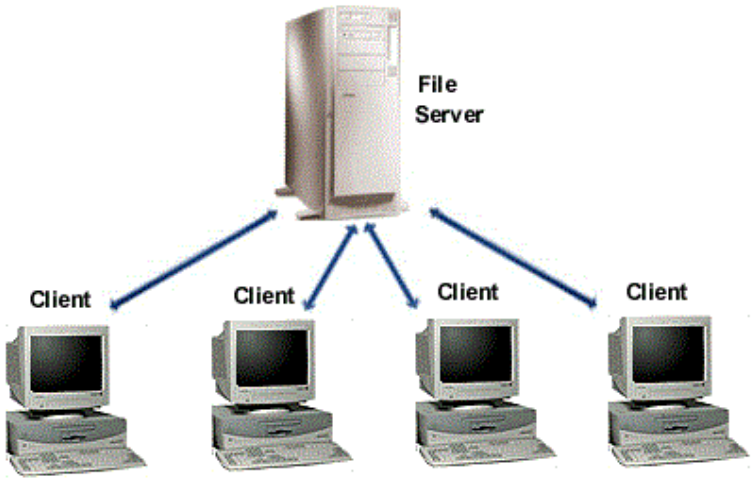


Fig. 1.8: नेटवर्क ऑपरेटिंग सिस्टम (Network Operating System)

नेटवर्क ऑपरेटिंग सिस्टमची उदाहरणे (Examples of Network Operating System)

1. मायक्रोसॉफ्ट विंडोज सर्व्हर 2003 (Microsoft Windows Server 2003)
2. मायक्रोसॉफ्ट विंडोज सर्व्हर 2008 (Microsoft Windows Server 2008)
3. यूनिक्स (UNIX)
4. लिनक्स (Linux)
5. मॅक ओएस एक्स (Mac OS X)

6. नोव्हेल नेटवेअर **(**Novell NetWare)
7. बीएसडी- बर्कले सॉफ्टवेअर डिस्ट्रिब्युशन **(**BSD- Berkeley Software Distribution)

नेटवर्क ऑपरेटिंग सिस्टमचे फायदे (Advantages of Network Operating System)

1. सेंट्रलाइझ्ड सर्व्हर (Centralized Server)अत्यंत स्थिर असते .
2. सुरक्षा (Security) सर्व्हर व्यवस्थापित करतो .
3. नवीन तंत्रज्ञान आणि हार्डवेअरमध्ये अपग्रेड करून सिस्टममध्ये सहजपणे इंटिग्रेटेड केले जाऊ शकते.
4. सर्व्हरवर वेगवेगळ्या ठिकानाहून आणि सिस्टमच्या प्रकारांमधून रिमोट एक्सेस (Remote Access) करणे शक्य आहे.

नेटवर्क ऑपरेटिंग सिस्टमचे तोटे (Disadvantages of Network Operating System)

1. सर्व्हर किंमत आणि रन करण्याची किंमत जास्त असते.
2. बहुतेक ऑपरेशन्ससाठी सेंट्रल लोकेशनवर अवलंबन असणे.
3. नियमित देखभाल आणि अपडेट्स करणे आवश्यक आहेत.

1.2.6 टाइम शेअर्ड किंवा टाइम शेअरिंग ऑपरेटिंग सिस्टम (Time-shared or Time Sharing operating system)

टाइम-शेअरिंग (Time Sharing) हे एक तंत्र आहे, जे एकाच वेळी विशिष्ट संगणक सिस्टिम वापरण्यास विविध टर्मिनलवर स्थित असलेल्या बऱ्याच लोकांना सक्षम करते. टाइम-शेअरिंग किंवा मल्टीटास्किंग हा मल्टीप्रोग्रामिंगचा लॉजिकल एक्स्टेंशन आहे. प्रोसेसरची वेळ (Processor's Time), जो एकाच वेळी एका पेक्षा अधिक यूजर्समध्ये शेअर केला जातो, त्याला टाइम-शेअरिंग (Time Sharing) म्हणून संबोधले जाते. मल्टी-प्रोग्रॅम्ड बॅच सिस्टम आणि टाइम-शेर्ड सिस्टममधील मुख्य फरक असा आहे की मल्टी-प्रोग्रामेड बॅच सिस्टमच्या बाबतीत, प्रोसेसरचा वापर जास्तीत जास्त करणे हे आहे, तर टाइम-शेर्ड सिस्टमचे उद्दीष्ट रिस्पॉन्स टाइम (Response Time) कमी करणे आहे. सीपीयू एका पेक्षा अधिक जॉब्सच्या दरम्यान स्विच (Switch) करून एक्झिक्युट करतो, परंतु स्विच वारंवार आढळतात. अशा प्रकारे, यूझरला त्वरित प्रतिसाद मिळू शकतो. उदाहरणार्थ, ट्रांझ्याक्शन प्रोससिंग (Transaction Processing) मध्ये, प्रोसेसर प्रत्येक यूजर प्रोग्राम थोड्या प्रमाणात बर्स्ट (Burst) किंवा कॉम्पुटेशनचा क्वांटममध्ये (Quantum of Computation) एक्झिक्युट करतो. म्हणजेच, जर एन (N) यूजर्स उपस्थित असतील तर प्रत्येक युजरला टाइम क्वांटम मिळू शकेल. जेव्हा यूजर कमांड (Command) सबमिट करतो, तेव्हा रिस्पॉन्स टाइम काही सेकंदात असतो. ऑपरेटिंग सिस्टम प्रत्येक युजरला वेळेचा एक छोटासा भाग (Time quantum) प्रदान करण्यासाठी सीपीयू शेड्यूलिंग आणि मल्टीप्रोग्रामिंग वापरते. प्रामुख्याने बॅच सिस्टम म्हणून डिझाइन केलेल्या संगणक प्रणाली टाइम-शेअरिंग सिस्टममध्ये सुधारणा केल्या गेल्या आहेत. टाइम शेअरिंग ऑपरेटिंग सिस्टम Fig. 1.9 मध्ये दर्शविली आहे

उदाहरण: विंडोज (Windows), लिनक्स (LINUX), युनिक्स (UNIX)

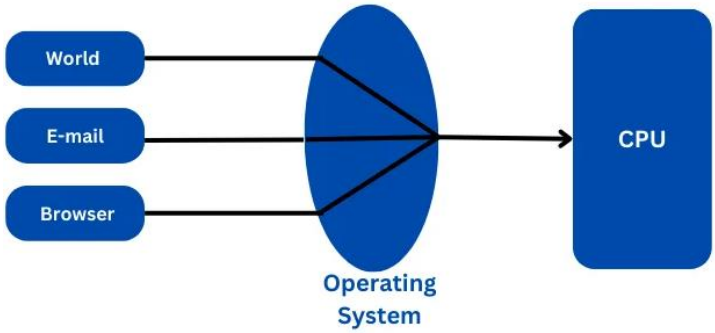


Fig. 1.9: **टाइम-शेअरिंग ऑपरेटिंग सिस्टम (Time Sharing Operating System)**

टाइम शेअरिंग ऑपरेटिंग सिस्टमचे फायदे (Advantages of Time Sharing Operating System)

1. प्रत्येक टास्कला (Task) समान संधी मिळते.
2. सॉफ्टवेअरच्या डुप्लिकेशनची शक्यता कमी.
3. सीपीयूची निष्क्रिय वेळ (Idle Time) कमी केला जाऊ शकतो.

टाइम शेअरिंग ऑपरेटिंग सिस्टमचे तोटे (Disadvantages of Time Sharing Operating System)

1. रिलायबिलिटीची (Reliability) समस्या
2. यूजर प्रोग्राम आणि डेटाच्या सुरक्षा (Security)आणि इंटिग्रिटीची (Intigirty) काळजी घेणे आवश्यक आहे.
3. यूजर प्रोग्राम आणि डेटाची डेटा कम्यूनिकेशनची समस्या.
4. डेटा कम्यूनिकेशनची समस्या.

1.2.7 रिअल टाइम सिस्टम (Real Time System)

रिअल-टाइम सिस्टमची व्याख्या डेटा प्रोसेसिंग सिस्टिम (Data Processing System**)** म्हणून केली जाते ज्यात इनपुटला प्रोसेस करण्यासाठी आणि रीस्पॉन्ड करण्यासाठीचा आवश्यक टाइम इंटर्वल इतके लहान असते की ते कॉम्पुटिंग इन्व्होर्मेन्ट (Computing Environment) नियंत्रित करते. इनपुटला प्रतिसाद देण्यासाठी आणि आवश्यक अपडेट केलेल्या माहिती प्रदर्शित करण्यासाठी सिस्टमला लागणारा वेळस रिस्पॉन्स टाइम (Response Time) असे म्हणतात**.** म्हणून या पद्धतीत**,** ऑनलाइन प्रोसेसच्या तुलनेत रिस्पॉन्स टाइम खूपच कमी असते. जेव्हा प्रोसेसरच्या ऑपरेशनवर रिजिड टाइमची (Rigit Time) आवश्यकता असते किंवा डेटा आणि रीअल-टाइम सिस्टमचा फ्लो डेडिकेटेड अप्लिकेशनचे डिव्हाइस कंट्रोलची गरज असते तेव्हा रीअल-टाइम सिस्टम वापरली जाते. रिअल-टाइम ऑपरेटिंग सिस्टममध्ये वेल डिफाइन्ड **,** निश्चित वेळेची मर्यादा असणे आवश्यक आहे**,** अन्यथा सिस्टम अयशस्वी होईल. उदाहरणार्थ**,** वैज्ञानिक प्रयोग**,** वैद्यकीय इमेजिंग सिस्टम**,** औद्योगिक नियंत्रण प्रणाली**,** शस्त्रे प्रणाली**,** रोबोट्स**,** एअर ट्रॅफिक कंट्रोल सिस्टम इ.

रिअल-टाइम ऑपरेटिंग सिस्टमचे दोन प्रकार आहेत.

1. हार्ड रीअल-टाइम सिस्टम (Hard real-time systems)

हार्ड रीअल-टाइम सिस्टम क्रिटिकल टास्क (Critical Task) वेळेवर पूर्ण करण्याची हमी देतात. हार्ड रीअल-टाइम सिस्टममध्ये**,** सेकंडरी स्टोरेज (Secondary Storage) मर्यादित किंवा नसतेच आणि डेटा रॉममध्ये (ROM) स्टोअर्ड केला जातो. या सिस्टममध्ये**,** व्हर्च्युअल मेमरी (Virtual Memory) जवळजवळ कधीही वापरत नाही.

2. सॉफ्ट रीअल-टाइम सिस्टम (Soft real-time systems)

सॉफ्ट रीअल-टाइम सिस्टम कमी रिस्ट्रिक्टिव (Ristrictive) असते. एक क्रिटिकल रीअल-टाइम टास्क इतर टास्क पेक्षा जास्त प्रायोरिटी (Priority) मिळवते आणि पूर्ण होईपर्यंत प्रायोरिटी टिकवून ठेवते. सॉफ्ट रीअल-टाइम सिस्टम हार्ड रीअल-टाइम सिस्टमपेक्षा मर्यादित उपयुक्तता असते उदाहरणार्थ मल्टीमीडिया (Multimedia), वर्चुअल रिऍलिटी (Virtual Reality), अंडर सी एक्सप्लोरेशन (Under Sea Exploration) आणि प्लॅनेटरी रोव्हर्स (Planetary Rovers), सारख्या ऍडवान्सड वैज्ञानिक प्रकल्प इ.

उदाहरणे: VxWorks**,** एम्बेडेड Linux, SAFERTOS, ThreadX

1.2.8 मोबाईल ओएस (अँड्रॉइड ओएस) (Mobile OS (Android OS))

मोबाइल ऑपरेटिंग सिस्टम ही ओएस आहेत जी विशेषत: स्मार्टफोन, टॅब्लेट आणि वेअरेबल डिव्हाइसना पॉवर करण्यासाठी डिझाइन केलेले असते. मोबाइल ऑपरेटिंग सिस्टम ही एक ऑपरेटिंग सिस्टम आहे जी मोबाइल डिव्हाइसवर इतर अप्लिकेशन सॉफ्टवेअर रन करण्यास मदत करते. हे लिनक्स (Linux) आणि विंडोज (Windows) सारख्या प्रसिद्ध संगणक ऑपरेटिंग सिस्टमसारखेच सॉफ्टवेअर आहे, परंतु आता ते काही प्रमाणात लाईट आणि सिम्पल असते. स्मार्टफोनमध्ये आढळणाऱ्या ऑपरेटिंग सिस्टीममध्ये सिम्बियन ओएस (Symbian OS), आयफोन ओएस (iPhone OS), रिमचे ब्लॅकबेरी (RIM's BlackBerry), विंडोज मोबाईल (Windows Mobile), पाम वेबओएस (Palm WebOS), अँड्रॉइड (Android) आणि मेमो (Maemo) यांचा समावेश आहे. अँड्रॉइड (Android), वेबओएस (WebOS) आणि मेमो (Maemo)हे सर्व लिनक्सपासून घेतलेले आहेत. आयफोन ओएसची उत्पत्ती बीएसडी (BSD) आणि नेक्स्टस्टेप (NeXTSTEP) पासून झाली आहे, जे युनिक्सशी (UNIX) संबंधित आहेत. हे संगणक आणि हातानी वापरण्याच्या डिव्हाइसचे सौंदर्य एकत्र करते. यात सामान्यत: टेलिफोनी आणि इंटरनेट कनेक्शनसाठी सेल्युलर बिल्ट-इन मॉडेम आणि सिम ट्रे असते. आपण मोबाइल खरेदी केल्यास, निर्माता कंपनी त्या विशिष्ट डिव्हाइससाठी ओएस निवडते. जगभरातील विविध मोबाइल प्लॅटफॉर्ममध्ये अँड्रॉइड ऑपरेटिंग सिस्टम (Android Operating System) हा सर्वात मोठा स्थापित बेस आहे. जगातील 190 हून अधिक देशांमध्ये कोट्यवधी मोबाइल डिव्हाइस अँड्रॉइडद्वारे समर्थित आहेत. अँड्रॉइड ही

लिनक्स कर्नल आणि इतर ओपन-सोर्स सॉफ्टवेअरच्या सुधारित आवृत्तीवर आधारित एक ऑपरेटिंग सिस्टम आहे, जी प्रामुख्याने स्मार्टफोन आणि टॅब्लेट सारख्या टचस्क्रीन-आधारित मोबाइल उपकरणांसाठी डिझाइन केलेली आहे

1.3 कमांड लाइन आधारित ऑपरेटिंग सिस्टम (Command line based Operating System)

कमांड लाइन इंटरफेस (CLI) हा एक टेक्स्ट-आधारित यूजर इंटरफेस (Text based user interface) आहे जो संगणक फायली पाहण्यासाठी आणि व्यवस्थापित (Manage) करण्यासाठी वापरला जातो. कमांड लाइन इंटरफेसना कमांड-लाइन यूजर इंटरफेस, कन्सोल यूजर इंटरफेस आणि कॅरेक्टर यूजर इंटरफेस असेही म्हणतात.

कमांड लाइन इंटरफेस (CLI) चा इतिहास आणि वैशिष्ट्ये

माउसच्या (Mouse) आधी, यूजर्स ऑपरेटिंग सिस्टम (ओएस) किंवा अप्लिकेशनशी कीबोर्ड (Keyboard)वापरून इंटरॅक्ट करत होते. युजर्सला संगणकावर टास्क (Task) रन करण्यासाठी कमांड लाइन इंटरफेसमध्ये कमांड टाइप केल्या जायच्या. थोडक्यात, कमांड लाइन इंटरफेसमध्ये पांढऱ्या टेक्स्ट सह ब्लॅक बॉक्स असायचा. कमांड टाइप करून यूजर्स कमांड लाइन इंटरफेसमधील प्रॉमप्टला रीस्पॉन्ड करतो. सिस्टमच्या आउटपुट किंवा सिस्टिमचा रिस्पॉन्स, टेबल, लिस्ट किंवा सिस्टम किंवा अप्लिकेशनचे ऍक्शन काही इतर कन्फर्मेशन समाविष्ट असू शकते.

कमांड बेस्ड ऑपरेटिंग सिस्टमचे उदाहरणः

1. बहुतेक सध्याच्या युनिक्स-आधारित सिस्टम्स कमांड लाइन इंटरफेस आणि ग्राफिकल यूजर इंटरफेस दोन्ही देतात.
2. विंडोज ऑपरेटिंग सिस्टममधील MS-DOS ऑपरेटिंग सिस्टम आणि कमांड शेल (Command Shell) ही कमांड लाइन इंटरफेसची उदाहरणे आहेत.
3. याव्यतिरिक्त, प्रोग्रामिंग भाषा कमांड लाइन इंटरफेसना समर्थन देऊ शकतात, जसे की पायथॉन (Python).

1.3.1 जियूआई आधारित ऑपरेटिंग सिस्टम (GUI based Operating System)

जीयूआय (ग्राफिकल यूजर इंटरफेस) संगणक सॉफ्टवेअरसाठी इंटरॅक्टिव्ह व्हिज्युअल कॉम्पोनन्ट्सची एक प्रणाली आहे. जीयूआय ऑब्जेक्टस दर्शविते जी माहित सांगतात आणि यूजरद्वारे घेतलेल्या क्रियांचे ऍक्शन करते. जेव्हा यूजर त्यांच्याशी इंटरॅक्ट करतो तेव्हा ऑब्जेक्ट्स रंग, आकार किंवा व्हिजीबिलटी बदलते. जीयूआय ऑब्जेक्ट्समध्ये चिन्ह, कर्सर आणि बटणे समाविष्ट असतात. हे ग्राफिकल घटक कधीकधी ध्वनी किंवा विझूअल इफेक्ट्स जसे कि ट्रान्सपॅरेंसी आणि ड्रॉप श्याडो सारख्या व्हिज्युअल इफेक्टसह वर्धित केले जातात. MS-DOS सारख्या टेक्स्ट-आधारित कमांड-लाइन इंटरफेस किंवा युनिक्स-सारख्या ऑपरेटिंग सिस्टमच्या शेलपेक्षा GUI हा अधिक यूजर फ्रेंडली (User-friendly) मानला जातो. GUI प्रथम झेरॉक्स पीएआरसी येथे अलन के (Alan Kay), डग्लस एंजेलबार्ट (Douglas Engelbart) आणि 1981 मध्ये इतर संशोधकांच्या गटाने विकसित केले. नंतर Apple पलने 19 जानेवारी 1983 रोजी जीयूआयसह लिसा संगणकाची (Lisa Computer) ओळख करुन दिली.

GUI ऑपरेटिंग सिस्टमची उदाहरणे

1. मायक्रोसॉफ्ट विंडोज (Microsoft Windows)
2. ॲपल सिस्टम 7 आणि मॅकओएस (Apple System 7 and M)
3. क्रोम ओएस (Chrome OS)
4. उबंटू सारखे लिनक्स प्रकार जीयूआय इंटरफेस वापरतात (Linux variants like Ubuntu using a GUI interface)

1.4 ऑपरेटिंग सिस्टमच्या विविध सेवा (Different Services of Operating System)

ऑपरेटिंग सिस्टम वा सिस्टम यूजर्सना आणि प्रोग्राम्सनाही वेगवेगळ्या प्रकारच्या सेवा (Serivces) प्रदान करते. ते ॲप्लिकेशन प्रोग्राम्स (जे ऑपरेटिंग सिस्टममध्ये चालतात) मुक्तपणे एक्झिक्युट करण्यासाठी एक वातावरण देखील प्रदान करते. ते यूजर्सना सोयीस्कर पद्धतीने विविध प्रोग्राम्स रन (Run) करण्यासाठी सेवा प्रदान करते. ऑपरेटिंग सिस्टमद्वारे प्रदान केलेल्या काही सामान्य सेवा खालीलप्रमाणे आहेत.

1. यूजर इंटरफेस (User Interface)
2. प्रोग्राम एक्झिक्युशन (Program Execution)

3. फाइल सिस्टम मॅनिपुलेशन (File system manipulation)
4. इनपुट / आउटपुट ऑपरेशन्स (Input / Output Operations)
5. कम्यूनिकेशन (Communication)
6. रिसोर्स अलोकेशन (Resource Allocation)
7. एरर डिटेक्शन (Error Detection)
8. अकाउंटिंग (Accounting)
9. सुरक्षा आणि संरक्षण (Security and protection)

1.4.1 यूजर इंटरफेस (User Interface)

सामान्यतः ऑपरेटिंग सिस्टम तीन प्रकारांमध्ये किंवा प्रकारांमध्ये येते. इंटरफेसनुसार त्यांचे प्रकार आणखी विभागले गेले आहेत आणि हे आहेत:

1. कमांड लाइन इंटरफेस **(**Command line interface-CLI)
2. बॅच आधारित इंटरफेस **(**Batch based interface)
3. ग्राफिकल यूजर इंटरफेस **(**Graphical User Interface)

कमांड लाइन इंटरफेस (**CLI):** सहसा टेक्स्ट कमांड आणि त्या कमांड एंटर करण्याच्या तंत्राचा वापर करते.

बॅच इंटरफेस (BI): कमांड आणि डायरेक्टिव्ह्जचा वापर फाइल्समध्ये एंटर केलेल्या कमांडचे मॅनेजमेंट करण्यासाठी केला जातो आणि त्या फाइल्स एक्झिक्युट होतात.

ग्राफिकल यूजर इंटरफेस (GUI): ही एक विंडो सिस्टम आहे ज्यामध्ये पॉइंटिंग डिव्हाइस (जसे की माउस किंवा ट्रॅकबॉल) असते जे I/O कडे निर्देशित करते, मेनूमधून इंटरफेस निवडते आणि कीबोर्डमधून मजकूर एंटर करण्यासाठी आणि अनेक लिस्ट पाहण्याचे पर्याय बनवते.

1.4.2 प्रोग्राम एक्झिक्युशन (Program Execution)

ऑपरेटिंग सिस्टममध्ये प्रोग्राम मेमरीमध्ये लोड करण्याची आणि त्या प्रोग्राम एक्झिक्युट करण्याची क्षमता असणे आवश्यक आहे. शिवाय, प्रोग्राम सामान्यत: किंवा असामान्य / जबरदस्तीने त्याचे एक्झिक्युशन समाप्त करण्यास सक्षम असणे आवश्यक आहे. प्रोग्राम रन करण्यासाठी, प्रोग्राम प्रथम रॅममध्ये लोड करणे आवश्यक आहे आणि नंतर त्याच्या एक्झिक्युशनसाठी सीपीयू टाइम देणे आवश्यक आहे. ऑपरेटिंग सिस्टम यूजर्सच्या सोयीसाठी हे कार्य करते. हे मेमरीचे अलोकेशन आणि डी-अलोकेशन, सीपीयू शेड्यूलिंग इ. यासारख्या इतर महत्त्वपूर्ण कार्ये देखील करते. प्रोग्राम मॅनेजमेन्ट संदर्भात ऑपरेटिंग सिस्टमच्या मुख्य ऍक्टिव्हिटीज खालीलप्रमाणे आहेत -

1. मेमरीमध्ये प्रोग्राम लोड करणे.
2. प्रोग्राम एक्झिक्युट करणे.
3. प्रोग्रामचे एक्झिक्युशन हाताळने
4. प्रोसेस सिंक्रोनाइझेशनसाठी एक मेक्यानिज्म प्रदान करते.
5. प्रोसेस कम्यूनिकेशनसाठी एक मेक्यानिज्म प्रदान करते.
6. डेडलॉक (Deadlock) हाताळणीसाठी एक मेक्यानिज्म प्रदान करते.

1.4.3 फाईल सिस्टिम मॅनिप्युलेशन (File system manipulation)

फाइल म्हणजे संबंधित माहितीचा संग्रह. संगणक दीर्घकालीन स्टोरेजसाठी डिस्कवर (सेकंडरी स्टोरेज) फायली साठवू शकतात. स्टोरेज मीडियाची उदाहरणे म्हणजे मॅग्नेटिक टेप मॅग्नेटिक डिस्क (Magnetic Disk) आणि सीडी (CD), डीव्हीडी (DVD) सारखे ऑप्टिकल डिस्क ड्राइव्ह. या प्रत्येक मीडियाचे स्वतःचे गुणधर्म आहेत जसे की वेग, क्षमता, डेटा ट्रान्सफर रेट आणि डेटा ॲक्सेस पद्धती. फाइल सिस्टम सामान्यतः सोप्या नेव्हिगेशन आणि वापरासाठी डिरेक्टरीज (Directories) मध्ये व्यवस्थित केली जाते. या डिरेक्टरीज मध्ये फायली आणि इतर डिरेक्टरीज असू शकतात.

फाइल व्यवस्थापना (File Management) संदर्भात ऑपरेटिंग सिस्टमच्या प्रमुख ऍक्टिव्हिटीज खालीलप्रमाणे आहेत -

1. प्रोग्रामला फाइल वाचण्याची (Read) किंवा फाइल लिहिण्याची (Write) आवश्यकता असते.
2. ऑपरेटिंग सिस्टम प्रोग्रामला फाइलवर ऑपरेशन करण्याची परवानगी देते.

3. पर्मिशन देणे जसे कि रीड ओन्ली (Read Only), रीड राईट (Read-Write), डिनाईड (Denied)आणि अशाच प्रकारे इतर.
4. ऑपरेटिंग सिस्टम युजर्सला फाइल्स क्रिएट (Create)/ डिलीट (Delete) साठी इंटरफेस प्रदान करते.
5. ऑपरेटिंग सिस्टम युजर्सला डिरेक्टरीज (Create)/ डिलीट (Delete) साठी इंटरफेस प्रदान करते.
6. ऑपरेटिंग सिस्टम फाइल सिस्टमचा बॅकअप (Backup) तयार करण्यासाठी इंटरफेस प्रदान करते.

1.4.4 इनपुट/आउटपुट ऑपरेशन्स (Input / Output Operations)

प्रत्येक प्रोग्रामला इनपुट आवश्यक असते णि प्रोसेस केल्यानंतर, यूजर्सने सबमिट केलेले इनपुटनुसार आउटपुट तयार करते. यात I/O डिव्हाइसचा वापर समाविष्ट आहे. ऑपरेटिंग सिस्टम आय/ओ डिवाइसचे हार्डवेअरच्या यूजर्स पासून लपवते. म्हणून ऑपरेटिंग सिस्टम यूजर्सना I/O फंक्शन्स प्रदान करून प्रोग्राम रन करण्यास सोयीस्कर करते. यूजर्स-लेव्हल प्रोग्राम आय/ओ सर्विस प्रदान करू शकत नाहीत आणि ते ऑपरेटिंग सिस्टमद्वारे प्रदान करणे आवश्यक आहे. ऑपरेटिंग सिस्टम यूजर आणि डिव्हाइस ड्रायव्हर्स (Device Driver) मधील संवाद व्यवस्थापित करते.

1. I/O ऑपरेशन म्हणजे कोणत्याही फाईल किंवा कोणत्याही विशिष्ट I/O डिव्हाइससह रीड (Read) किंवा राईट (Write) ऑपरेशन.
2. ऑपरेटिंग सिस्टम आवश्यक असल्यास आवश्यक I/O डिव्हाइसचा एक्सेस प्रदान करते..

1.4.5 कम्यूनिकेशन (Communication)

ऑपरेटिंग सिस्टम शेअर्ड मेमरीच्या स्वरूपात विविध प्रकारच्या प्रोसेस मध्ये कम्यूनिकेशन करते. मल्टीटास्किंग एनव्होर्मेन्ट मध्ये, प्रक्रियेस एकमेकांशी कम्यूनिकेशन करणे आणि त्यांची माहिती देवाणघेवाण करणे आवश्यक आहे. कम्यूनिकेशनच्या संदर्भात ऑपरेटिंग सिस्टमच्या मुख्य ऍक्टिव्हिटीज खालीलप्रमाणे आहेत -

1. दोन प्रोसेस बहुतेकदा त्या दरम्यान डेटा हस्तांतरित (Data Transfer) करणे आवश्यक असते
2. दोन्ही प्रोसेस एका संगणकावर किंवा वेगवेगळ्या संगणकावर असू शकतात परंतु संगणक नेटवर्कद्वारे कनेक्ट केल्या असतात.
3. शेअर्ड मेमरीद्वारे किंवा मेसेज पासिंगद्वारे कम्यूनिकेशन केले जाऊ शकते.

1.4.6 रिसोर्स अलोकेशन (Resource Allocation)

मल्टीटास्किंग एनव्होर्मेन्ट मध्ये, जेव्हा एकाच वेळी अनेक जॉब्स चालू असतात, तेव्हा प्रत्येक प्रोसेसचा चांगल्या प्रकारे वापर करण्यासाठी आवश्यक रिसोर्स (जसे की CPU, मुख्य मेमरी, टेप ड्राइव्ह किंवा सेकंडरी स्टोरेज इ.) वाटप करणे ही ऑपरेटिंग सिस्टमची जबाबदारी असते. रिसोर्स मॅनेजमेंट संदर्भात ऑपरेटिंग सिस्टमच्या मुख्य ऍक्टिव्हिटीज खालीलप्रमाणे आहेत -

1. ओएस शेड्यूलर्सचा (Schedulars) वापर करून सर्व प्रकारच्या रिसोर्सचे मॅनेजमेंट करते.
2. सीपीयू शेड्यूलिंग अल्गोरिथम (Scheduling Algorithm) सीपीयूच्या चांगल्या वापरासाठी वापरले जातात.

1.4.7 एरर डिटेक्शन (Error Detection)

सीपीयू (CPU), मेमरी हार्डवेअर (Memory Hardware), आय/ओ डिव्हाइसेस (I/O Deices) आणि यूजर प्रोग्राममध्ये एरर येऊ शकतात. प्रत्येक प्रकारच्या एरर साठी, ऑपरेटिंग सिस्टम योग्य आणि सुसंगत कॉम्पुटिंग सुनिश्चित करण्यासाठी पुरेशी कारवाई करते. ऑपरेटिंग सिस्टम हार्डवेअर समस्यांना देखील हाताळते. हार्डवेअर समस्या टाळण्यासाठी, ऑपरेटिंग सिस्टम एरर शोधण्यासाठी आणि या एरर (आढळल्यास) दुरुस्त करण्यासाठी सिस्टमचे सतत निरीक्षण करते. ऑपरेटिंग सिस्टमचे मुख्य कार्य म्हणजे हार्ड डिस्कवरील बॅड सेक्टर (Bad Sector), मेमरी ओव्हरफ्लो (Memory overflow) आणि आय/ओ डिव्हाइसेसशी संबंधित एरर शोधणे. एरर शोधल्यानंतर, ऑपरेटिंग सिस्टम सुसंगत कॉम्पुटिंग साठी योग्य कारवाई करते.

1.4.8 अकाउंटिंग (Accounting)

ऑपरेटिंग सिस्टम प्रत्येक प्रोसेसद्वारे किंवा यूजरद्वारे वापरल्या जाणाऱ्या सर्व संसाधनांचा (Resource) हिशेब ठेवते. मल्टीटास्किंगमध्ये (Multitasking), अकाउंटिंग प्रत्येक प्रोसेससाठी संसाधनांचे वाटप करून सिस्टमची कार्यक्षमता वाढवते ज्यामुळे प्रत्येक प्रोसेसचे समाधान सुनिश्चित होते.

1.4.9 सुरक्षा आणि संरक्षण (Security and protection)

जर संगणक सिस्टिममध्ये अनेक यूजर्स असतील आणि अनेक प्रोसेसच्या कॉन्करंट एक्झिक्युशनची परवानगी असेल तर विविध प्रोसेस एकमेकांच्या ऍक्टिव्हिटीज पासून संरक्षित (Protect) केल्या पाहिजेत. संरक्षणामध्ये नियंत्रित पद्धतीने सिस्टम संसाधनांमध्ये सर्व एक्सेस सुनिश्चित करणे समाविष्ट असते. सिस्टम सुरक्षित (Secure) करण्यासाठी, यूजर्सला वापरण्यापूर्वी किंवा सिस्टममध्ये प्रमाणीकृत (Authenticate) करणे आवश्यक असते.

1.5 सिस्टिम कॉल्स (System Calls)

1.5.1 संकल्पना (Concept)

सिस्टम कॉल ऑपरेटिंग सिस्टम सर्व्हिसेसना इंटरफेस प्रदान करते. प्रोसेस आणि ऑपरेटिंग सिस्टम दरम्यानचा इंटरफेस सिस्टम कॉलद्वारे प्रदान केला जातो. सर्वसाधारणपणे, सिस्टम कॉल असेंब्ली लँग्वेज इंस्ट्रक्शन्स मध्ये उपलब्ध असतात. जेव्हा यूजर मोडमधील प्रोसेसला रिसोर्सचा एक्सेसची आवश्यकता असते तेव्हा सहसा सिस्टम कॉल्स केले जातात. मग ते कर्नलला (Kernel) सिस्टम कॉलद्वारे संसाधन प्रदान करण्यासाठी विनंती करते. जेव्हा यूजर मोडमधील (User mode) प्रोग्रामला रॅम किंवा हार्डवेअर रिसोर्सचा एक्सेस आवश्यक असतो, तेव्हा त्या रिसोर्सचा एक्सेस प्रदान करण्यासाठी कर्नलला विचारणे आवश्यक असते. जे सिस्टम कॉल द्वारे केले जाते. जेव्हा एखादा प्रोग्राम सिस्टम कॉल करतो, तेव्हा यूजर मोडमधून कर्नल मोडवर (Kernel Mode) स्विच केला जातो. याला कॉन्टेक्स्ट स्विच (Context Switch) म्हणतात. नंतर कर्नल प्रोग्रामने विनंती केलेले रिसोर्स प्रदान करतो. त्यानंतर, आणखी एक कॉन्टेक्स्ट स्विच होतो ज्यामुळे मोड कर्नल मोडमधून पुन्हा युजर मोडमध्ये बदलतो. साधारणपणे, यूजर स्तरावरील प्रोग्रामद्वारे सिस्टम कॉल खालील परिस्थितींमध्ये केले जातात:

1. फाइल सिस्टममध्ये फाइल्स तयार करणे, उघडणे (Open), बंद करणे (Close) आणि हटवणे (Delete).
2. नवीन प्रोसेस तयार करणे (Create) आणि व्यवस्थापित (Manage) करणे.
3. नेटवर्कमध्ये कनेक्शन तयार करणे, पॅकेट पाठवणे (Sent) आणि प्राप्त (Receive) करणे.
4. माऊस किंवा प्रिंटर सारख्या हार्डवेअर डिव्हाइसवर एक्सेस ची विनंती करणे.

सिस्टम कॉलच्या एक्झिक्युशनचे प्रतिनिधित्व करणारी आकृती Fig. 1.10 दिली आहे:

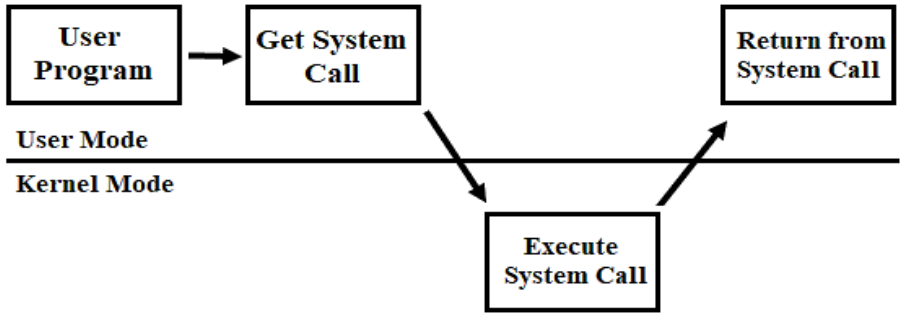


Fig. 1.10: ऑपरेटिंग सिस्टिममधील सिस्टिम कॉल्स (System calls in Operating System)

सिस्टम कॉलद्वारे प्रदान केलेल्या सेवा:

1. प्रोसेस निर्मिती आणि मॅनेजमेंट (Process Creation and Management)
2. मुख्य मेमरी मॅनेजमेंट (Main Memory Management)
3. फाइल एक्सेस, डिरेक्टरी आणि फाइल सिस्टम मॅनेजमेंट
4. I/O डिव्हाइस हाताळणी
5. संरक्षण (Protection)
6. नेटवर्किंग इ.

1.5.2 सिस्टम कॉल्सचे प्रकार (Types of System Calls)

सिस्टम कॉल्सच्या 5 खाली दिलेल्या वेगवेगळ्या श्रेणी आहेत.

a. प्रोसेस नियंत्रण (Process control)

ऑपरेटिंग सिस्टममध्ये प्रक्रिया तयार करण्यासाठी आणि व्यवस्थापित करण्यासाठी प्रोसेस कंट्रोल सिस्टम कॉलचा वापर केला जातो. प्रोसेस कंट्रोल सिस्टम कॉलचे कार्य खालीलप्रमाणे आहे:

1. जेव्हा आवश्यक असेल तेव्हा ते प्रोसेस तयार (Create) करते आणि समाप्त (Delete) करते.
2. ते चालू असलेल्या प्रोग्रामचे सामान्यपणे समाप्त (End) किंवा असामान्यपणे रद्द करणे (Abort) देखील नियंत्रित करते.
3. प्रोग्राम एक्झिक्युट करताना प्रोसेस कंट्रोल सिस्टम कॉल दुसऱ्या प्रोग्रामचे लोडिंग आणि एक्झिक्युशन नियंत्रित करते.
4. प्रोसेस एक्झिक्युशन दरम्यान स्विच करताना ते प्रोसेसचे ऍट्रिब्यूट (Attribute) निर्धारित करते आणि रीसेट करते.
5. ते प्रोसेसला मेमरी वाटप करण्यास मदत करते आणि प्रोसेस समाप्त (Terminate) झाल्यावर मेमरी देखील मुक्त करते.
6. ते प्रोसेस एक्झिक्युशन पूर्ण करण्यासाठी आवश्यक असलेल्या प्रतीक्षा वेळेवर (Waiting Time) नियंत्रण ठेवते.
7. प्रोसेस नियंत्रण कॉल्स सिस्टिम कॉल करते जे प्रोसेसच्या एक्झिक्युशन पूर्ण झाल्याचे संकेत देतात.

b. फाइल मॅनेजमेंट (File management)

फाइल मॅनेजमेंट सिस्टम कॉल्स संगणकाच्या फाइल आणि डायरेक्टरी सिस्टमवर नियंत्रण ठेवतात.

1. हे कॉल्स सिस्टमच्या फाइल्स तयार (Create) आणि डिलीट करतो.
2. हे कॉल्स एक नवीन फाइल तयार करतो आणि ती डिलीट देखील नियंत्रित करतो.
3. हे कॉल्स फाइल उघडण्यासाठी (Open) आणि बंद (Close) करण्यासाठी आवश्यक आहे.
4. कॉल्सचा वापर फाइलवर वाचन (Read) आणि लेखन (Write) ऑपरेशन्स करण्यासाठी देखील केला जातो.
5. कॉल्सचा वापर फाईल्सचे ऍट्रिब्यूट (Attribute) निश्चित करण्यासाठी आणि रीसेट (Reset) करण्यासाठी देखील आवश्यक आहे.

c. डिव्हाइस मॅनेजमेंट (Device management)

संगणक प्रणालीशी जोडलेल्या डिव्हाइसचे नियंत्रण आणि मॅनेजमेंट करण्यासाठी डिव्हाइस मॅनेजमेंट सिस्टिम कॉल आवश्यक असते.

1. हे कॉल्स सिस्टमच्या अनेक डिव्हाइसेससाठी प्रोसेसच्या विनंत्या (Requests) नियंत्रित करतो.
2. हे कॉल्स डिव्हाइसचे वाचन (Read), लेखन (Write) ऑपरेशन्स नियंत्रित करते.
3. ते प्रोसेस एक्झिक्युशन करण्यासाठी संसाधने (Resources) प्रदान करते आणि प्रक्रिया पूर्ण झाल्यानंतर डिव्हाइसेस देखील रिलीज करते.

d. इन्फॉर्मशन मेंटेनन्स (Information maintenance)

इन्फॉर्मशन मेंटेनन्स सिस्टिम कॉल संगणकाची माहिती जसे की सिस्टम वेळ (System time), तारीख (Date) इत्यादी राखतो.

1. हे कॉल सिस्टमची माहिती मिळविण्यासाठी वापरले जातात जसे की ऑपरेशन सिस्टमची आवृत्ती **(Version),** सिस्टमचे कॉन्फिगरेशन.
2. ऑपरेटिंग सिस्टमच्या सर्व प्रोसेस बद्दल माहिती मिळविण्यासाठी.
3. सिस्टमवरील सर्व फायलींबद्दल माहिती मिळविण्यासाठी.
4. सिस्टमशी जोडलेल्या सर्व डिव्हाइसेस बद्दल माहिती मिळविण्यासाठी.

e. कम्युनिकेशन (Communications)

तुमच्या संगणकाला नेटवर्कवर जोडण्यासाठी एक कम्युनिकेशन कॉल आवश्यक आहे.

1. ते नेटवर्कवर कम्युनिकेशन कनेक्शन तयार करते आणि डिलीट करते.
2. नेटवर्कवरून संदेश पाठवण्यासाठी आणि प्राप्त करण्यासाठी देखील हा कॉल आवश्यक असते.
3. नेटवर्कवरून कनेक्ट केलेल्या डिव्हाइसेसची स्थिती माहिती हस्तांतरित करते.
4. नेटवर्कवरून तुमच्या संगणकाशी कनेक्ट केलेले रिमोट डिव्हाइस जोडण्यासाठी किंवा वेगळे करण्यासाठी देखील हा कॉल आवश्यक असते.

Table 1.1: विंडोज (Windows) आणि युनिक्स (UNIX) सिस्टम कॉलची उदाहरणे

सिस्टम कॉल्स (System Calls)	विंडोज **(Windows)**	युनिक्स **(Unix)**
प्रोसेस नियंत्रण (Process Control)	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
फाइल मॅनेजमेंट (File management)	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
डिव्हाइस मॅनेजमेंट (Device management)	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
इन्फॉर्मशन मेंटेनन्स (Information maintenance)	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
कम्युनिकेशन (Communications)	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
संरक्षण (Protection)	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

1.6 ऑपरेटिंग सिस्टमचे घटक (Components of Operating System)

ऑपरेटिंग सिस्टम ही एक मोठी आणि गुंतागुंतीची सिस्टिम आहे जी फक्त लहान भागांमध्ये विभागून तयार केली जाऊ शकते. हे भाग सिस्टमचा एक सुव्यवस्थित भाग असले पाहिजेत, इनपुट, आउटपुट आणि फंक्शन्स काळजीपूर्वक परिभाषित करतात. जरी विंडोज (Windows), मॅक (Mach), युनिक्स (UNIX), लिनक्स (Linux)आणि इतर ऑपरेटिंग सिस्टम्समध्ये समान रचना नसली तरी, बहुतेक ऑपरेटिंग सिस्टम्समध्ये समान ओएस सिस्टम घटक असतात, जसे की फाइल, मेमरी, प्रोसेस, आय/ओ डिव्हाइस मॅनेजमेंट (Management). ऑपरेटिंग सिस्टमचे घटक विविध संगणक सिस्टमचे भाग एकत्र काम करण्यासाठी महत्त्वाची भूमिका बजावतात. ऑपरेटिंग सिस्टमचे खालील घटक आहेत, जसे की:

1. प्रोसेस मॅनेजमेंट (Process Management)
2. फाइल मॅनेजमेंट (File Management)
3. नेटवर्क मॅनेजमेंट (Network Management)
4. मुख्य मेमरी मॅनेजमेंट (Main Memory Management)
5. सेकण्डरी स्टोरेज मॅनेजमेंट (Secondary Storage Management)
6. आय/ओ डिव्हाइस मॅनेजमेंट (I/O Device Management)
7. सुरक्षा मॅनेजमेंट (Security Management)
8. कमांड इंटरप्रिटर सिस्टम (Command Interpreter System)

Fig. 1.11 मध्ये दाखवलेले ऑपरेटिंग सिस्टम घटक तुम्हाला CPU आणि मेमरी हार्डवेअर त्रुटी शोधून योग्य संगणन मिळविण्यात मदत करतात.

Fig.1.11: ऑपरेटिंग सिस्टमचे घटक (Components of Operating System)

1.6.1 प्रोसेस मॅनेजमेंट (Process Management)

प्रोसेस मॅनेजमेंट घटक ही ऑपरेटिंग सिस्टमवर एकाच वेळी चालणाऱ्या अनेक प्रोसेस व्यवस्थापित करण्याची प्रोसेस आहे. प्रत्येक एक्झिक्युशन चालू असलेल्या सॉफ्टवेअर ॲप्लिकेशन प्रोग्राममध्ये त्यांच्याशी संबंधित एक किंवा अधिक प्रोसेस असतात. प्रोसेस मॅनेजमेंट प्रक्रिया कार्यक्षमतेने चालू ठेवते. ते त्यांना वाटप केलेल्या मेमरीचा वापर करते आणि आवश्यकतेनुसार त्या बंद करते. प्रोसेसचे एक्झिक्युशन अनुक्रमिक असली पाहिजे जेणेकरून प्रोसेसच्या वतीने किमान एक इंस्ट्रक्शन एक्झिक्युट केली पाहिजे. Fig.1.12 प्रोसेस मॅनेजमेंट दर्शवते.

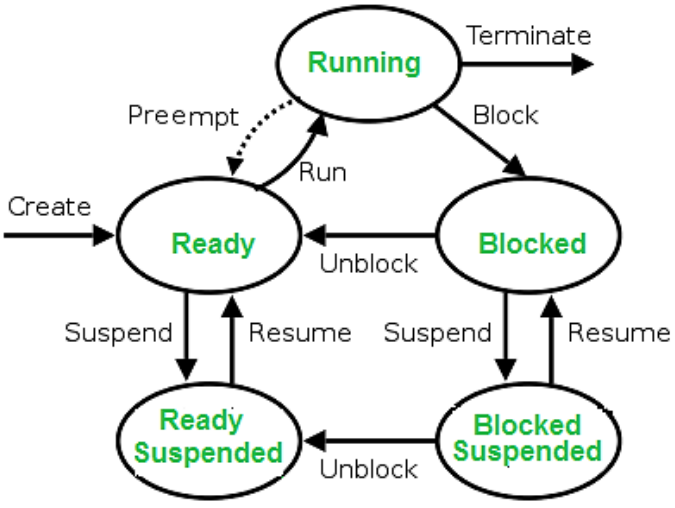


Fig.1.12: प्रोसेस मॅनेजमेंट (Process Management)

प्रोसेस व्यवस्थापनाची कार्ये (Function of Process Management)

ऑपरेटिंग सिस्टममध्ये प्रोसेस व्यवस्थापनाची खालील कार्ये आहेत, जसे की:

1. प्रोसेस तयार करणे (Create) आणि हटवणे (Delete).
2. प्रोसेस निलंबन (Suspension) आणि पुनरारंभ (Resumption).
3. प्रोसेस सिंक्रोनाइझेशन
4. प्रोसेस कम्युनिकेशन
5. डेडलॉक (Deadlock) हाताळणी

1.6.2 फाइल मॅनेजमेंट (File Management)

फाईल त्याच्या निर्मात्याने परिभाषित केलेल्या संबंधित माहितीचा एक संच आहे. हे सामान्यत: प्रोग्राम्स (सोर्स (Source) आणि ऑब्जेक्ट (Object) फॉर्म दोन्ही) आणि डेटा दर्शवते. डेटा फायली वर्णमाला (Alphabetic), संख्यात्मक (Numeric) किंवा अल्फान्यूमेरिक (Alphanumeric) असू शकतात. . Fig.1.13 फाईल मॅनेजमेंट दर्शवते.

ऑपरेटिंग सिस्टममध्ये फाइल व्यवस्थापनाच्या संदर्भात खालील महत्त्वपूर्ण ऍक्टिव्हिटीज आहेत:

1. फाइल आणि डिरेक्टरी निर्मिती (Creation) आणि हटविणे (Deletion).
2. फायली आणि डिरेक्टरी हाताळण्यासाठी.
3. सेकंडरी स्टोरेजवर फायलीचे मॅपिंग.
4. स्टेबल स्टोरेज मीडियावर फाईल बॅकअप करणे.

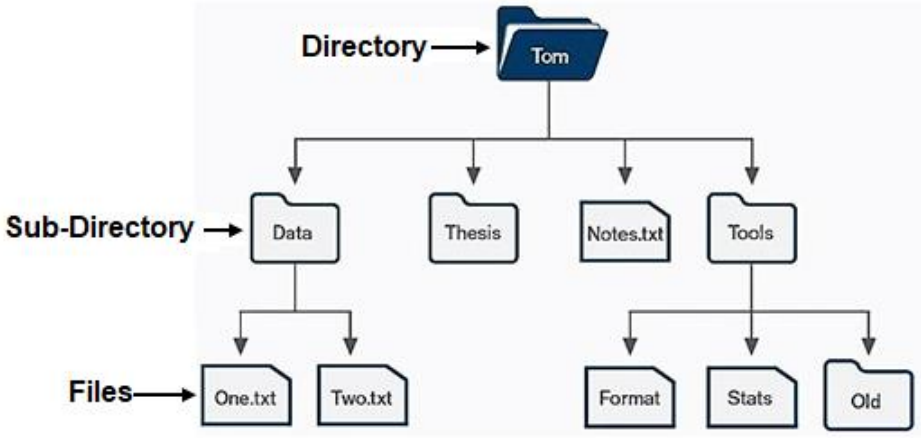


Fig.1.13: फाइल मॅनेजमेंट (File Management)

1.6.3 नेटवर्क मॅनेजमेंट (Network Management)

नेटवर्क मॅनेजमेंट ही संगणक नेटवर्क प्रशासन (Administration) आणि व्यवस्थापित (Managing) करण्याची प्रक्रिया आहे. यात परफॉर्मन्स मॅनेजमेंट, नेटवर्कची तरतूद, फॉल्ट विश्लेषण (Fault analysis) आणि सेवेची गुणवत्ता (Quality of service) राखणे समाविष्ट आहे. डिस्ट्रीब्युटेड सिस्टिम ही संगणक किंवा प्रोसेसरचा संग्रह आहे जी कधीही त्यांची मेन मेमरी आणि क्लॉक (Clock) शेअर करीत नाहीत. या प्रकारच्या सिस्टममध्ये, सर्व प्रोसेसरकडे त्यांची स्थानिक मेमरी (Local Memory) असते आणि प्रोसेसर फायबर ऑप्टिक्स किंवा टेलिफोन लाइन सारख्या वेगवेगळ्या कम्युनिकेशन केबल्सचा वापर करून एकमेकांशी संवाद साधतात. नेटवर्कमधील संगणक कम्युनिकेशन नेटवर्कद्वारे कनेक्ट केलेले असतात, जे बऱ्याच वेगवेगळ्या प्रकारे कॉन्फिगर करू शकतात. नेटवर्क व्यवस्थापनात नेटवर्क पूर्णपणे किंवा अंशतः कनेक्ट होऊ शकते, जे यूजर्सना कनेक्शन आणि सुरक्षिततेच्या समस्यांवर मात करणारे मार्ग (Routing) आणि कनेक्शन रणनीती डिझाइन करण्यात मदत करते.

नेटवर्क व्यवस्थापनाची कार्ये (Function of Network Management)

नेटवर्क मॅनेजमेंट खालील कार्ये प्रदान करते, जसे की:

1. डिस्ट्रीब्युटेड सिस्टिम आपल्याला आकार आणि फंक्शनमधील विविध संगणकीय संसाधनांमध्ये मदत करतात. त्यामध्ये मिनीकॉम्पुटर्स, मायक्रोप्रोसेसर आणि बऱ्याच जनरल पर्पज कॉम्पुटरचा समावेश असू शकतो.
2. डिस्ट्रीब्युटेड सिस्टिम नेटवर्कच्या विविध संसाधनांमध्ये युजर्सला एक्सेस देखील देते.
3. शेअर्ड संसाधनांला एक्सेस करण्यास मदत करते जे कॉम्पुटेशन वेगवान करण्यासाठी किंवा डेटा उपलब्धता आणि विश्वसनीयता प्रदान करण्यास मदत करते.

1.6.4 मुख्य मेमरी मॅनेजमेंट (Main Memory Management)

मुख्य मेमरी हा स्टोरेज किंवा बाइटचा एक मोठा ॲरे आहे, ज्याला एक ॲड्रेस (Address) असतो. विशिष्ट मेमरी ॲड्रेस च्या वाचन (Reading) किंवा लिहिण्याच्या (Writing) अनुक्रमांचा वापर करून मेमरी मॅनेजमेंट प्रक्रिया आयोजित केली जाते. ते ऍबसॉल्यूट अड्रेसवर (Absolute Address) मॅप केलेले असले पाहिजे आणि प्रोग्राम एक्झिक्युट करण्यासाठी प्रोग्राम मेमरीत लोड केले जाते. मेमरी मॅनेजमेंट पद्धतीची निवड अनेक घटकांवर अवलंबून असते. तथापि, हे प्रामुख्याने

सिस्टमच्या हार्डवेअर डिझाइनवर आधारित असते. प्रत्येक अल्गोरिथमला (Algorithm) संबंधित हार्डवेअर सपोर्ट आवश्यक आहे. मुख्य मेमरी वेगवान स्टोरेज ऑफर करते जी सीपीयू थेट एक्सेस करू शकते. मुख्य मेमरी महाग असते आणि म्हणूनच स्टोरेज क्षमता कमी असते. तथापि, प्रोग्राम एक्झिक्युट करण्यासाठी, तो मुख्य मेमरीमध्ये असणे आवश्यक आहे. स्वॅपिंग (Swapping) म्हणजे मुख्य मेमरीमधून सेकंडरी मेमरीमध्ये तात्पुरते प्रोसेस स्वॅप करण्याची प्रक्रिया, जी सेकंडरी मेमरीच्या तुलनेत वेगवान असते. स्वॅपिंग अनेक प्रोसेसेस रन करण्याची अनुमती देते आणि Fig.1.14 मध्ये दर्शविल्याप्रमाणे एकाच वेळी मेमरीमध्ये फिट होऊ शकते.

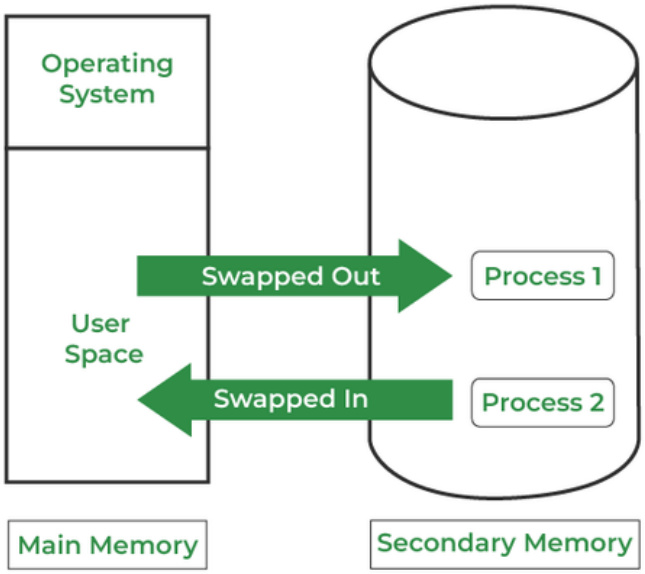


Fig.1.14: मुख्य मेमरी मॅनेजमेंट (Main Memory Management)

मेमरी व्यवस्थापनाची कार्ये (Function of Main Memory Management)

ऑपरेटिंग सिस्टम मेमरी व्यवस्थापनासाठी खालील कार्ये करते:

1. हे आपल्याला प्रायमरी मेमरीचा मागोवा (Track) ठेवण्यास मदत करते.
2. त्याचा कोणता भाग कुठल्या प्रोसेसकडून वापरात आहे आणि कोणता भाग वापरात नाही हे ठरवते.
3. मल्टीप्रोग्रामिंग सिस्टममध्ये, ओएस कोणत्या प्रोसेसला मेमरी मिळेल आणि किती मिळेल हे ठरवते.
4. जेव्हा प्रोसेस मेमरीसाठी विनंती करते तेव्हा मेमरीचे वाटप करते.
5. जेव्हा एखाद्या प्रोसेसला यापुढे मेमरीची आवश्यकता नसते किंवा प्रोसेस संपुष्टात (Terminated) आणले जाते तेव्हा ते मेमरीला डी-अलोकेट (De-Allocate) करते.

1.6.5 सेकण्डरी स्टोरेज मॅनेजमेंट (Secondary Storage Management)

संगणक सिस्टिमचे सर्वात महत्वाचे कार्य म्हणजे प्रोग्राम्स एक्झिक्युट करणे. हे प्रोग्राम आपल्याला एक्झिक्युशन दरम्यान मुख्य मेमरीमधून डेटामध्ये एक्सेस करण्यास मदत करतात. संगणकाची ही मेमरी सर्व डेटा आणि प्रोग्राम कायमस्वरुपी स्टोअर करण्यासाठी खूपच लहान असते. संगणक सिस्टिम मुख्य मेमरीचा बॅक अप घेण्यासाठी सेकंडरी स्टोरेजचा वापर करते. सेकंडरी स्टोरेज मॅनेजमेंट Fig.1.15 मध्ये दर्शविले आहे.

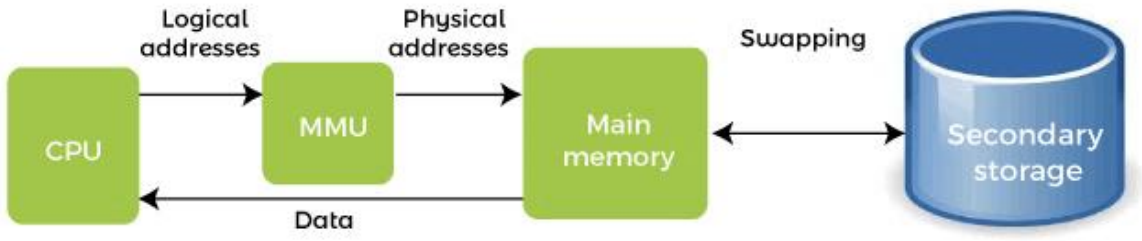


Fig.1.15: सेकण्डरी स्टोरेज मॅनेजमेंट (Secondary Storage Management)

आज आधुनिक संगणक हार्ड ड्राइव्ह/एसएसडीचा (Hard Drive/SSD) वापर प्रोग्राम आणि डेटा दोन्हीचा प्राथमिक स्टोरेज म्हणून वापरतात. तथापि, सेकंडरी स्टोरेज मॅनेजमेंट यूएसबी फ्लॅश ड्राइव्ह आणि सीडी/डीव्हीडी ड्राइव्हसारख्या

स्टोरेज डिव्हाइससाठी देखील केले जाते. असेंबलर्स (Assembler) आणि कंपाईलर (Compiler) सारखे प्रोग्राम डिस्कवर मेमरीमध्ये लोड होईपर्यंत साठवले जातात आणि नंतर डिस्कचा वापर प्रोसेसिंगसाठी सोअर्स आणि डेस्टिनेशन म्हणून केला जातो.

सेकंडरी स्टोरेज व्यवस्थापनाची कार्ये (Function of Secondary Storage Management)

ऑपरेटिंग सिस्टममध्ये सेकंडरी स्टोरेज व्यवस्थापनाची काही प्रमुख कार्ये खाली दिलेले आहेत:

1. सेकंडरी स्टोरेजचे वाटप (Secondary storage allocation)
2. मोकळा जागाचे मॅनेजमेंट (Free Space Management)
3. डिस्क शेड्यूलिंग (Disk Scheduling)

1.6.6 आय/ओ डिव्हाइस मॅनेजमेंट (I/O Device Management)

ऑपरेटिंग सिस्टमचे एक महत्त्वाचे कार्य म्हणजे माउस, कीबोर्ड, टच पॅड,ॲडॉप्टर्स, यूएसबी डिव्हाइस, नेटवर्क कनेक्शन, ऑडिओ I/O, प्रिंटर, डिस्क ड्राइव्हसह विविध आय/ओ डिव्हाइस व्यवस्थापित करणे. आय/ओ डिव्हाइस मॅनेजमेंट सिस्टिम प्राथमिकतेवर (Priority) आधारित प्रोसेसना इनपुट/आउटपुट डिव्हाइसचे वाटप करते आणि अटींवर अवलंबून तात्पुरते किंवा कायमस्वरुपी डी-अलोकेट (De-Allocate) करते. प्रोसेस बऱ्याच संसाधनांची मागणी करू शकते आणि जर संसाधने उपलब्ध नसतील तर आवश्यक संसाधने उपलब्ध होईपर्यंत प्रोसेसचे एक्झिक्युशन थांबविले जाते आणि जर संसाधने उपलब्ध असतील तर त्या प्रोसेसला अलोकेट (Allocate) केली जातात. ऑपरेटिंग सिस्टम आणि हार्डवेअर किंवा संगणकाची डिव्हाइस थेट एकमेकांशी कनेक्ट केलेली नसतात, जी डिव्हाइस ड्रायव्हर्स (Device Drivers) म्हणून ओळखल्या जाणाऱ्या विशेष प्रोग्रामद्वारे एकमेकांशी जोडल्या जातात. डिव्हाइस ड्रायव्हर्सचे मुख्य कार्य म्हणजे ऑपरेटिंग सिस्टमची हाय लेव्हल प्रोग्रामिंग लँग्वेज आणि हार्डवेअर डिव्हाइसच्या इलेक्ट्रिक सिग्नल दरम्यान अनुवादक (Translator) म्हणून कार्य करणे. आय/ओ डिव्हाइस मॅनेजमेंट Fig.1.16 मध्ये दर्शविले आहे.

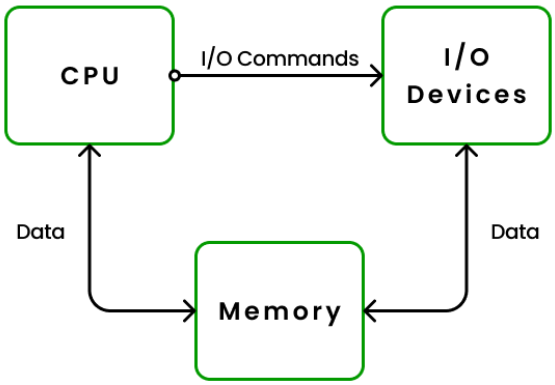


Fig.1.16: आय/ओ डिव्हाइस मॅनेजमेंट (I/O Device Management)

I/O व्यवस्थापनाची कार्ये **(Function of Secondary Storage Management)**

I/O मॅनेजमेंट प्रणाली खालील कार्ये करते, जसे की:

1. बफर कॅशिंग सिस्टम (Buffer Caching System) देते
2. सामान्य डिव्हाइस ड्रायव्हर (Device Driver) कोड प्रदान करते
3. विशिष्ट हार्डवेअर डिव्हाइसेससाठी डिव्हाइस ड्रायव्हर्स प्रदान करते.
4. I/O तुम्हाला विशिष्ट डिव्हाइसची वैयक्तिकता (Individualities) जाणून घेण्यास मदत करते.

1.6.7 सेक्युरिटी मॅनेजमेंट (Security Management)

ऑपरेटिंग सिस्टममधील विविध प्रक्रिया इतर ऍक्टिव्हिटीज पासून सुरक्षित करणे आवश्यक आहे. म्हणून, विविध यंत्रणा हे सुनिश्चित करू शकतात की ज्या प्रोसेसेस फाइल्स, मेमरी, सीपीयू आणि इतर हार्डवेअर संसाधने ऑपरेट करू इच्छितात त्यांना ऑपरेटिंग सिस्टमकडून योग्य परवानगी असावी. सुरक्षा म्हणजे प्रोग्राम्स, प्रक्रिया किंवा यूजर्सना संगणक नियंत्रणांद्वारे परिभाषित केलेल्या संसाधनांमध्ये ऍक्सेस नियंत्रित करण्यासाठीची एक यंत्रणा. उदाहरणार्थ, मेमरी ॲड्रेसिंग हार्डवेअर प्रोसेस त्याच्या स्वतःच्या ॲड्रेस स्पेसमध्ये एक्झिक्युट केली जाऊ शकते याची पुष्टी करण्यास मदत करते. वेळ हे सुनिश्चित करते की कोणत्याही प्रोसेसचा अंत केल्याशिवाय CPU वर नियंत्रण नाही. शेवटी, कोणत्याही प्रोसेसला स्वतःचे

I/O संरक्षण करण्याची परवानगी नसते, जे तुम्हाला विविध पेरिफेरल डिवाइसची इंटेग्रिटी राखण्यास मदत करते. कॉम्पोनंट सबसिस्टिम मधील इंटरफेसमध्ये सुप्त त्रुटी शोधून सुरक्षा विश्वासार्हता सुधारू शकते. इंटरफेस त्रुटींचे लवकर निदान केल्याने खराब झालेल्या सबसिस्टिमद्वारे हेल्दी सबसिस्टिमची मलफंक्शनिंग रोखता येते. असुरक्षित संसाधनाचा गैरवापर अनधिकृत किंवा अक्षम यूजर्सद्वारे केला जाऊ शकत नाही.

1.6.8 कमांड इंटरप्रिटर सिस्टम (Command Interpreter System)

ऑपरेटिंग सिस्टमच्या सर्वात महत्वाच्या घटकांपैकी एक म्हणजे त्याचा कमांड इंटरप्रिटर. कमांड इंटरप्रिटर हा वापरकर्ता आणि उर्वरित सिस्टममधील प्राथमिक इंटरफेस आहे जो Fig.1.17 मध्ये दर्शविला आहे.

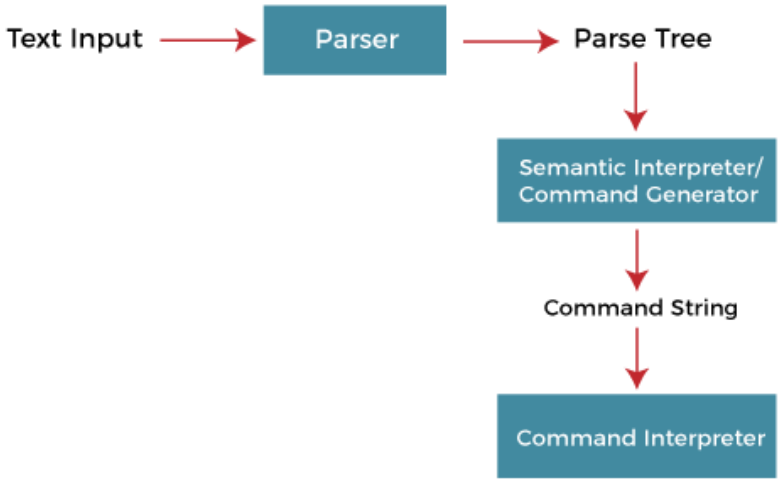


Fig. 1.17: कमांड इंटरप्रिटर सिस्टम (Command Interpreter System)

ऑपरेटिंग सिस्टमला कंट्रोल स्टेटमेंट्सद्वारे अनेक कमांड दिले जातात. बॅच सिस्टममध्ये नवीन जॉब सुरू झाल्यावर किंवा यूजर टाइम-शेअर्ड सिस्टममध्ये लॉग इन केल्यावर कंट्रोल स्टेटमेंट्स वाचणारा आणि त्याचा अर्थ लावणारा प्रोग्राम स्वयंचलितपणे एक्झिक्युट होतो. या प्रोग्रामला वेगवेगळ्या नावांनी ओळखले जाते.

1. कंट्रोल कार्ड इंटरप्रिटर,
2. कमांड-लाइन इंटरप्रिटर,
3. शेल (Shell) युनिक्समध्ये, आणि असेच.

त्याचे कार्य अगदी सोपे आहे, पुढील कमांड स्टेटमेंट मिळवा आणि ते एक्झिक्युट करा. कमांड स्टेटमेंट्स प्रोसेस मॅनेजमेंट, I/O हाताळणी, सेकंडरी स्टोरेज मॅनेजमेंट, मेन मेमरी मॅनेजमेंट, फाइल सिस्टम ऍक्सेस, संरक्षण आणि नेटवर्किंगशी संबंधित असतात.

References:

- Operating System: A Concept-Based Approach by Dhananjay M. Dhamdhere, McGraw Hill Education 3rd edition, ISBN: 978-1259005589
- Operating Systems : Internals and Design Principles by William Stallings, Pearson Education 9th Edition, ISBN: 978-9352866717
- Linux The Complete Reference by Richard Petersen, McGraw Hill, 6th edition, ISBN: 978-0071492478
- Linux command line and shell scripting by Richard Blum, Wiley India, ISBN: 978-1118983843
- Operating System Concepts by Abraham Silberschatz and James Peterson, Wiley India, ISBN: 9781119454083

Websites:

1. https://www.geeksforgeeks.org/
2. https://www.tutorialspoint.com/
3. https://www.scaler.com/

E-learning material (Video lectures)

1. https://www.youtube.com/watch?v=3Qfx4geYN9I –For Overview of OS
2. https://www.youtube.com/watch?v=vBURTt97EkA –For Overview of OS
3. https://www.youtube.com/watch?v=QhRPNO2f0g0 -–For Overview of OS

युनिट 2
प्रोसेस मॅनेजमेंट
(Process Management)

विषय निष्पत्ती (Course Outcome)

CO 2: ऑपरेटिंग सिस्टममध्ये प्रोसेस मॅनेजमेंटच्या वेगवेगळ्या पैलूंचे वर्णन करा.

घटक निष्पत्ती (Theory Learning Outcome-TLO):

1. प्रोसेसची वेगवेगळे स्टेट्स (States) समजावून सांगा.
2. पीसीबी (प्रोसेस कंट्रोल ब्लॉक-Process Control Block) मधील प्रोसेस स्टॅकच्या वेगवेगळ्या घटकांच्या कार्ये वर्णन करा.
3. एकमेकांना हस्तक्षेप न करता एका पेक्षा अधिक प्रोसेस शेअर्ड केलेल्या संसाधनांमध्ये ऍक्सेस कसे करतात स्पष्ट करा.
4. मल्टीथ्रेडिंग मॉडेलची तुलना करा.

2.1 प्रोसेस (Process) व्याख्या (Definition)

प्रोसेस म्हणजे प्रोग्राम एक्झिक्युशन मध्ये आहे किंवा एक्झिक्युशनचे इन्स्टन्स (Instance) आहे. प्रोग्रामचे एक्झिक्युशन अनुक्रमिक फॅशनमध्ये प्रगती करणे आवश्यक आहे.

- प्रोसेस हि प्रोग्राम कोड सारखी नाही तर त्यापेक्षा बरेच काही आहे.
- प्रोसेस ही एक 'ऍक्टिव्ह' (Active) प्रक्रिया आहे जी प्रोग्रामच्या विरूद्ध आहे जी 'प्यासीव्ह' (Passive) प्रक्रिया मानली जाते.
- प्रोसेसच्या ऍट्रिब्यूटमध्ये हार्डवेअर स्टेट (Hardware State), मेमरी (Memory), सीपीयू (CPU) इ. समाविष्ट असतात.

कार्यक्षम कार्य करण्यासाठी प्रोसेस मेमरी चार विभागांमध्ये विभागली गेली असते जी Fig. 2.1 मध्ये दर्शविली आहे:

- **टेक्स्ट विभाग (Text Section)** कॅम्पाइल्ड (Compiled) प्रोग्राम कोडचा बनलेला असतो, जेव्हा प्रोग्राम लॉन्च केला जातो तेव्हा तो नॉन-वोलाटाईल स्टोरेजमधून (Non-Volatile Storage) रीड केला जातो.
- **डेटा विभाग (Deta Section)** ग्लोबल (Global) आणि स्टॅटिक (Static) व्हेरिएबल्सचा बनलेला असतो, मेन प्रोग्राम एक्झिक्युट करण्यापूर्वी अलोकेट (Allocate)आणि इनिशिअलाइज (Initialize) केला जातो.
- **हिप (Heap)** ही डायनॅमिक मेमरी अलोकेशनसाठी वापरली जाते आणि नवीन (New), डिलीट (Delete), मॅलोक (Malloc), फ्री (Free) इ. च्या कॉलद्वारे व्यवस्थापित केला जातो.
- **स्टॅक (Stack)** हि लोकल (Local) व्हेरिएबल्ससाठी वापरला जातो. स्टॅकवरील जागा जेव्हा घोषित (Declared) केली जाते तेव्हा स्थानिक व्हेरिएबल्ससाठी राखीव असते.

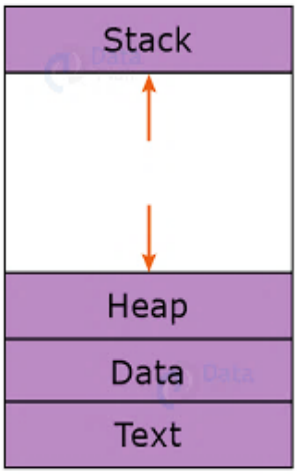


Fig. 2.1: प्रोसेस मेमरी (Process Memory)

2.1.1 प्रोसेस स्टेट्स (Process States)

जेव्हा एखादी प्रोसेस एक्झिक्युट होते तेव्हा ती वेगवेगळ्या स्टेट्स (States) किंवा स्टेजेस मधून जाते. हे स्टेजेस वेगवेगळ्या ऑपरेटिंग सिस्टममध्ये वेगवेगळे असू शकतात. सर्वसाधारणपणे, प्रोसेसची एका वेळी पुढील पाच स्टेट्स पैकी एक असू शकते.

1. **स्टार्ट (Start):** जेव्हा प्रोसेस प्रथम प्रारंभ (Start) केली जाते/तयार केली जाते तेव्हा ही प्रारंभिक स्टेट (State) असते.
2. **रेडी (Ready):** प्रोसेस प्रोसेसरला असाईन्ड करण्याची प्रतीक्षा करीत असते. रेडी प्रोसेस (Ready Process) ऑपरेटिंग सिस्टमद्वारे त्यांना प्रोसेसर अलोकेट करण्याची प्रतीक्षा करीत असते जेणेकरून प्रोसेस रन होऊ शकतील. प्रारंभ स्टेट नंतर किंवा प्रोसेस रन होत असताना जर शेड्युलरने (Schedular) दुसऱ्या प्रोसेसला सीपीयू असाइन करण्यासाठी इंटरप्ट (Interrupt) केले असल्यास, प्रोसेस या स्टेट मध्ये येऊ शकते.
3. **रनिंग (Running) :** एकदा ओएस शेड्यूलरद्वारे (OS Schedular) प्रोसेसरला प्रोसेस असाईन्ड केली गेली की प्रोसेसची स्टेट 'रनिंग' (Running) सेट केली जाते आणि प्रोसेसर त्या प्रोसेसचे इंस्ट्रक्शन्स एक्झिक्युट करायला सुरवात करतो.
4. **वेटिंग (Waiting):** प्रक्रिया वेटिंग स्टेटमध्ये जाते जर एखाद्या संसाधनाची प्रतीक्षा करण्याची आवश्यकता असेल, जसे की युझर्सच्या इनपुटची प्रतीक्षा करणे किंवा फाईल उपलब्ध होण्याची प्रतीक्षा करणे.
5. **टर्मिनेटेड किंवा एक्झिट (Terminated or Exit):** एखाद्या प्रोसेसचे एक्झिक्युशन संपल्यानंतर किंवा ऑपरेटिंग सिस्टमद्वारे ती टर्मिनेट करण्यात आली की ती टर्मिनेट स्टेट मध्ये हलविली जाते आणि ती मुख्य मेमरीमधून काढण्याची प्रतीक्षा करते.

प्रोसेस स्टेट डायग्रॅम Fig. 2.2 मध्ये दर्शविली आहे.

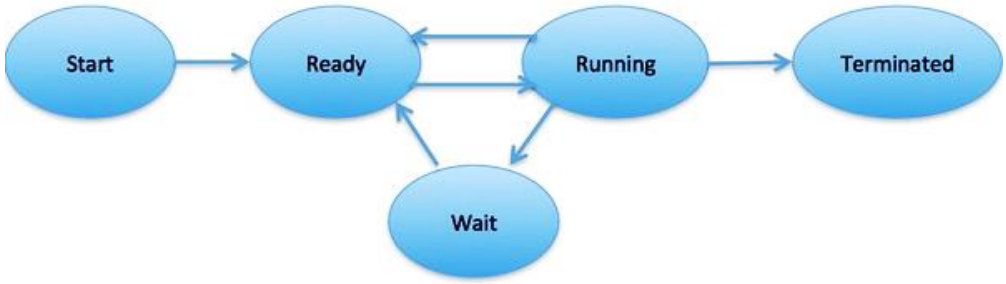


Fig. 2.2: प्रोसेस स्टेट डायग्रॅम (Process State Diagram)

2.1.2 प्रोसेस कंट्रोल ब्लॉक (Process Control Block (PCB))

प्रोसेस कंट्रोल ब्लॉक (Process Control Block) ही प्रत्येक प्रोसेससाठी ऑपरेटिंग सिस्टमद्वारे मेन्टेन केलेला डेटा स्ट्रक्चर (Data Structure) असतो. पीसीबी इंटीजर प्रोसेस आयडी (पीआयडी) (Process ID-PID) द्वारे ओळखला जातो. पीसीबी (PCB) आवश्यक ती सर्व माहिती टेबलमध्ये सूचीबद्ध केल्यानुसार प्रोसेसचा मागोवा ठेवण्यासाठी केली जाते जसे कि

1. **प्रोसेस स्टेट (Process State):** प्रोसेसची सद्य स्टेट म्हणजेच ती रेडी (Ready)आहे, रनिंग (Running) , वेटिंग (Waiting) किंवा जे काही असेल.
2. **प्रोसेस प्रिव्हिलेज (Process Privilege):** सिस्टमचे संसाधन ऍक्सेस करण्यास परवानगी देणे/काढणे आवश्यक असते त्यासाठी प्रोसेसची प्रिव्हिलेज लागते.
3. **प्रोसेस आयडी (Process ID) :** ऑपरेटिंग सिस्टममधील प्रत्येक प्रोसेससाठी युनिक इडेंटिफिकेशन .
4. **पॉईंटर (Pointer):** प्यारेन्ट (Parent) प्रोसेसचा पॉईंटर.
5. **प्रोग्राम काउंटर (Program Counter):** प्रोग्राम काउंटर या प्रोसेससाठी एक्झिक्युट होणाऱ्या पुढील इंस्ट्रक्शनचा ॲड्रेस चा मेमरी पॉईंटर आहे.
6. **सीपीयू रजिस्टर्स (CPU Registers):** विविध सीपीयू रेजिस्टर जे रनिंग स्टेटमध्ये प्रोसेसचा एक्झिक्युशनसाठी स्टोअर करणे आवश्यक असते.

7. **सीपीयू शेड्यूलिंग इन्फॉर्मेशन (CPU Scheduling Information):** प्रोसेसचे शेड्यूलिंग करण्यासाठी आवश्यक असलेली प्रायोरिटी (Priority) आणि इतर शेड्यूलिंग माहितीवर प्रक्रिया करते.
8. **मेमरी मॅनेजमेंट इन्फॉर्मेशन (Memory management information):** यात पेज टेबल (Page Table), मेमरी लिमिट (Limit), ऑपरेटिंग सिस्टमद्वारे वापरल्या जाणाऱ्या मेमरीवर अवलंबून सेगमेंट (Segment) टेबलची माहिती समाविष्ट असते.
9. **अकाउंटिंग इन्फॉर्मेशन (Accounting Information)**: यात प्रोसेस एक्झिक्युशनसाठी वापरल्या जाणाऱ्या सीपीयूची वेळ मर्यादा, एक्झिक्युशन आयडी इ. समाविष्ट असते.
10. **आयओ स्टेटस इन्फॉर्मेशन (IO status information)**: यात प्रोसेसला अलोकेट केलेल्या आय/ओ (I/O) डिव्हाइसची यादी समाविष्ट असते.

पीसीबीची एक सरलीकृत आकृती Fig. 2.3 मध्ये दर्शविली आहे.

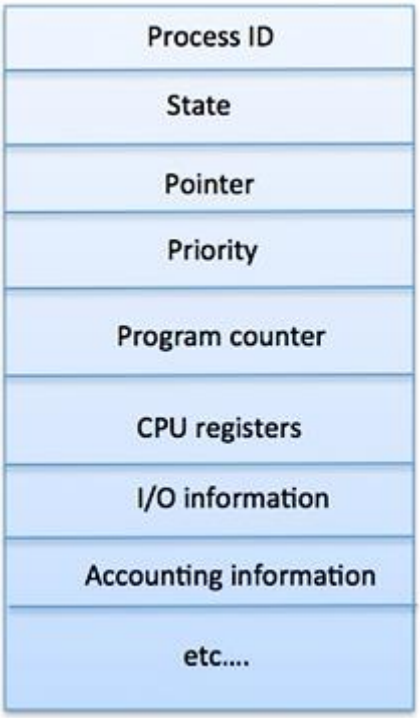


Fig. 2.3: प्रोसेस कंट्रोल ब्लॉक (Process Control Block (PCB))

पीसीबी त्याच्या प्रोसेसचा संपूर्ण लाईफ टाइमपर्यंत राखला जातो आणि एकदा प्रक्रिया संपल्यानंतर डिलीट केली जाते.

2.2 प्रोसेस शेड्यूलिंग (Process Scheduling)

प्रोसेस शेड्यूलिंग ही प्रोसेस मॅनेजमेंटची क्रिया आहे जी सीपीयूमधून चालू असलेली प्रोसेस काढून टाकते आणि विशिष्ट रणनीतीच्या आधारे दुसऱ्या प्रोसेसची निवड करते. प्रोसेस शेड्यूलिंग हा मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टमचा एक आवश्यक भाग आहे. अशा ऑपरेटिंग सिस्टम एकावेळी एक्झिक्युटेबल मेमरीमध्ये एकापेक्षा जास्त प्रोसेस लोड करण्याची परवानगी देते आणि लोड केलेली प्रोसेस टाइम मल्टिप्लेक्सिंगचा वापर करून सीपीयू शेअर करते.

2.2.1 शेड्यूलिंग क्यू (Scheduling Queues)

सिस्टममध्ये प्रवेश केल्यावर सर्व प्रोसेस, जॉब क्यूमध्ये (Job Queue) साठविल्या जातात. रेडी स्टेटमधील प्रोसेस रेडी क्यूमध्ये (Ready Queue) ठेवल्या जातात. डिव्हाइस उपलब्ध होण्याच्या प्रतीक्षेत असलेल्या प्रोसेस डिव्हाइस क्यूत (Device Queue) ठेवल्या जातात. प्रत्येक आय/ओ (I/O) डिव्हाइससाठी युनिक डिव्हाइस क्यू उपलब्ध असतात. सुरुवातीला नवीन प्रोसेस रेडी क्यूत ठेवली जाते. एक्झिक्युशनसाठी निवडल्या जाईपर्यंत हे रेडी क्यूत राहते. एकदा प्रोसेस सीपीयूला दिली गेली आणि एक्झिक्युट होत असताना, खालीलपैकी एक घटना घडू शकते जे मध्ये Fig. 2.4 दर्शिविली आहे.

- प्रोसेस आय/ओ विनंती (I/O Request) जारी करू शकते आणि नंतर आय/ओ क्यूत (I/O Queue) ठेवली जाऊ शकते.

- प्रोसेस एक नवीन सब-प्रोसेस तयार करू शकते आणि त्याच्या टर्मिनेशनची प्रतीक्षा करू शकते.
- इंटरप्टमुळे (Interrupt) प्रोसेस सीपीयूमधून जबरदस्तीने काढली जाऊ शकते आणि पुन्हा रेडी क्यूत (Ready Queue) ठेवली जाऊ शकते

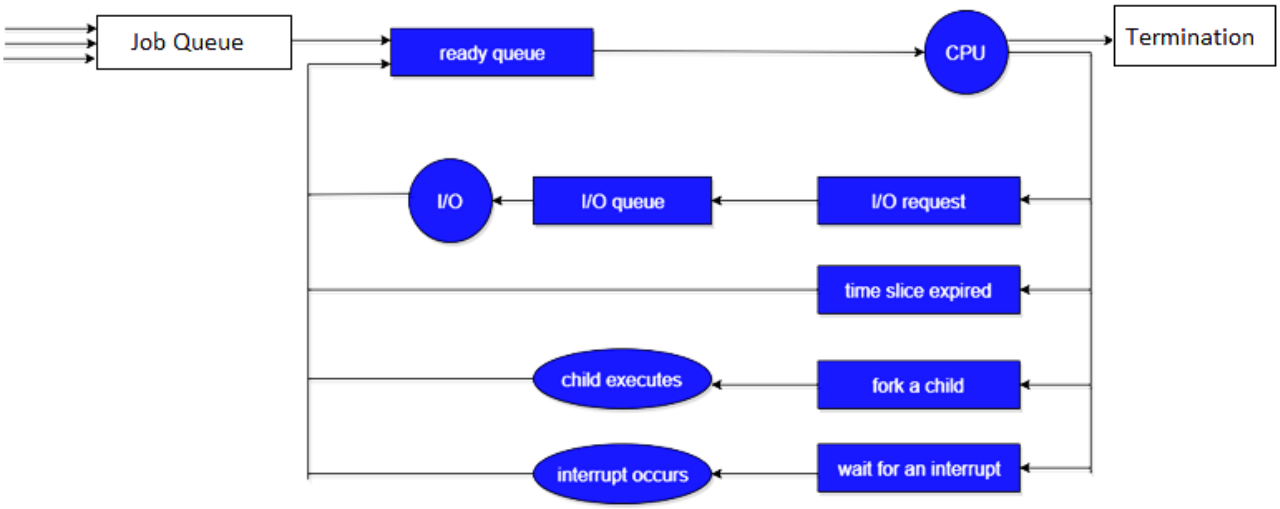


Fig. 2.4: प्रोसेसचे शेड्यूलिंग (Scheduling of Processes)

पहिल्या दोन प्रकरणांमध्ये, ही प्रोसेस अखेरीस वेटिंग स्टेटपासून रेडी स्टेटमध्ये स्विच करते आणि नंतर पुन्हा रेडी क्यूत ठेवली जाते. हें चक्र संपुष्टात येईपर्यंत ही प्रोसेस चालू ठेवते, ज्या वेळी ती सर्व क्यूतून काढली जाते त्यावेळी त्या प्रोसेसची पीसीबी (PCB) आणि अलोकेट (Allocate) केलेली संसाधने डीअलोकेट (De-allocate) केली जाते.

2.2.2 शेड्यूलर्सचे प्रकार (Types of Schedulers)

शेड्यूलर्स हे विशेष सिस्टम सॉफ्टवेअर आहेत जे प्रोसेस शेड्यूलिंग विविध प्रकारे हाताळतात. त्यांचे मुख्य कार्य म्हणजे सिस्टममध्ये सबमिट करण्यासाठी जॉब्स निवडणे आणि कोणती प्रोसेस रन करायची हे ठरविणे.

शेड्यूलर्स तीन प्रकारचे आहेत –

1. लॉन्ग टर्म शेड्यूलर (Long-Term Scheduler)
2. शॉर्ट टर्म शेड्यूलर (Short-Term Scheduler)
3. मिडीयम टर्म शेड्यूलर (Medium-Term Scheduler)

1. लॉन्ग टर्म शेड्यूलर (Long-Term Scheduler)

लॉन्ग टर्म शेड्यूलरला जॉब शेड्यूलर (Job Schedular) देखील म्हणतात. लॉन्ग टर्म शेड्यूलर प्रोसेसिंग साठी कोणता प्रोग्राम सिस्टममध्ये सबमिट करायचा हे निर्धारित करते. हे शेड्यूलर जॉब क्यूमधून प्रोसेसेस निवडते आणि एक्झिक्युशनसाठी मेमरीमध्ये लोड करते. सीपीयू शेड्यूलिंगसाठी मेमरीमध्ये प्रोसेस लोड होते. जॉब शेड्यूलरचे प्राथमिक उद्दीष्ट म्हणजे आय/ओ बाउंड (I/O bound) आणि प्रोसेसर बाउंड (Processor bound) सारख्या जॉबचे संतुलित मिश्रण प्रदान करणे. हे मल्टीप्रोग्रामिंगची डिग्री (Degree of Multiprogramming) देखील नियंत्रित करते. जर मल्टीप्रोग्रामिंगची डिग्री स्थिर असेल तर प्रोसेसच्या निर्मितीचा सरासरी रेट सिस्टम सोडणाऱ्या प्रोसेसच्या सरासरी डिपार्चरच्या रेट बरोबरीचा असणे आवश्यक आहे. काही सिस्टमवर, लॉन्ग टर्म शेड्यूलर उपलब्ध नसतो किंवा कमीतकमी असतो. टाइम शेअरिंग (Time Sharing) ऑपरेटिंग सिस्टममध्ये लॉन्ग टर्म शेड्यूलर नसतो. जेव्हा एखादी प्रोसेस नवीन (New) ते रेडी (Ready) मध्ये स्टेट बदलते, तेव्हा लॉन्ग टर्म शेड्यूलरचा वापर होतो.

2. शॉर्ट टर्म शेड्यूलर (Short-Term Scheduler)

शॉर्ट टर्म शेड्यूलरला सीपीयू शेड्यूलर (CPU Schedular) देखील म्हणतात. निवडलेल्या निकषांच्या संचाच्या अनुषंगाने सिस्टमची कार्यक्षमता वाढविणे हे त्याचे मुख्य उद्दीष्ट आहे. प्रोसेसची स्टेट रेडी (Ready) स्टेट ते रनिंग (Running) स्टेट मध्ये बदलली जाते. सीपीयू शेड्यूलर एक्झिक्युट होण्यास रेडी असलेल्या प्रोसेसमधील एक प्रोसेस निवडते आणि त्यापैकी एकास सीपीयू अलोकेट करते. शॉर्ट टर्म शेड्यूलर्स, ज्याला डिस्पॅचर्स म्हणून देखील ओळखले जाते, पुढील कोणत्या प्रोसेसला एक्झिक्युट करायचे याचा निर्णय घेते. शॉर्ट टर्म शेड्यूलर्स लॉन्ग टर्म शेड्यूलर्सपेक्षा वेगवान असतात.

3. मिडीयम टर्म शेड्यूलर (Medium-Term Scheduler)

मिडीयम टर्म शेड्यूलर शेड्यूलर स्वॅपिंगचा (Swapping) एक भाग आहे. हे मेमरीमधून प्रोसेस काढून टाकते आणि मल्टीप्रोग्रामिंगची डिग्री कमी करते. मिडीयम टर्म शेड्यूलर स्वॅप केलेल्या आउट-प्रोसेस हाताळण्याचे प्रभारी असतो. आय/ओ विनंती (I/O Request) केल्यास रनिंग (Running) असलेली प्रोसेस सस्पेंड (Suspend) होऊ शकते. सस्पेंडेड प्रोसेस पूर्ण होण्याच्या दिशेने कोणतीही प्रगती करू शकत नाही. या स्टेटत, मेमरीमधून प्रोसेस काढून टाकण्यासाठी आणि इतर प्रोसेससाठी जागा तयार करण्यासाठी, सस्पेंडेड प्रोसेस सेकंडरी स्टोरेजमध्ये हलविली जाते. या प्रक्रियेस स्वॅपिंग (Swapping) म्हणतात आणि ही प्रोसेस स्वॅप्ड-आऊट (Swapped Out) केली किंवा रोल्ड आऊट (Rolled Out) केली असे म्हणतात.

2.2.3 कॉन्टेक्स्ट स्विच (Context Switch)

कॉन्टेक्स्ट स्विच ही प्रोसेस कंट्रोल ब्लॉकमध्ये सीपीयूची स्टेट (State) किंवा कॉन्टेक्स्ट (Context) स्टोअर (Store) करण्याची आणि रिस्टोअर (Restore) करण्याची यंत्रणा आहे जेणेकरून नंतर त्याच बिंदूपासून प्रोसेस एक्झिक्युशन पुन्हा सुरू करता येते. या तंत्राचा वापर करून, कॉन्टेक्स्ट स्विचर (Context Switcher) एका पेक्षा अधिक प्रोसेसना एकच सीपीयू शेअर करण्यास सक्षम करते. कॉन्टेक्स्ट स्विचिंग हा मल्टीटास्किंग ऑपरेटिंग सिस्टम वैशिष्ट्यांचा एक आवश्यक भाग आहे. जेव्हा शेड्यूलर सीपीयूला एका प्रोसेसच्या एक्झिक्युशन पासून दुसऱ्या प्रोसेसच्या एक्झिक्युशनमध्ये स्विच करतो, तेव्हा रनिंग प्रोसेसची स्टेट (State) प्रोसेस कंट्रोल ब्लॉकमध्ये (PCB) स्टोअर केली जाते. यानंतर, पुढील प्रोसेससाठीची स्टेट (State) त्याच्या स्वतःच्या पीसीबीवरून लोड केली जाते आणि पीसी (PC), रजिस्टर्स (Registers) इत्यादी सेट करण्यासाठी वापरली जाते. त्या टप्प्यावर, दुसरी प्रोसेस एक्झिक्युट करण्यास सुरुवात होते. कॉन्टेक्सट स्विचिंग Fig. 2.5 मध्ये दर्शविले आहे

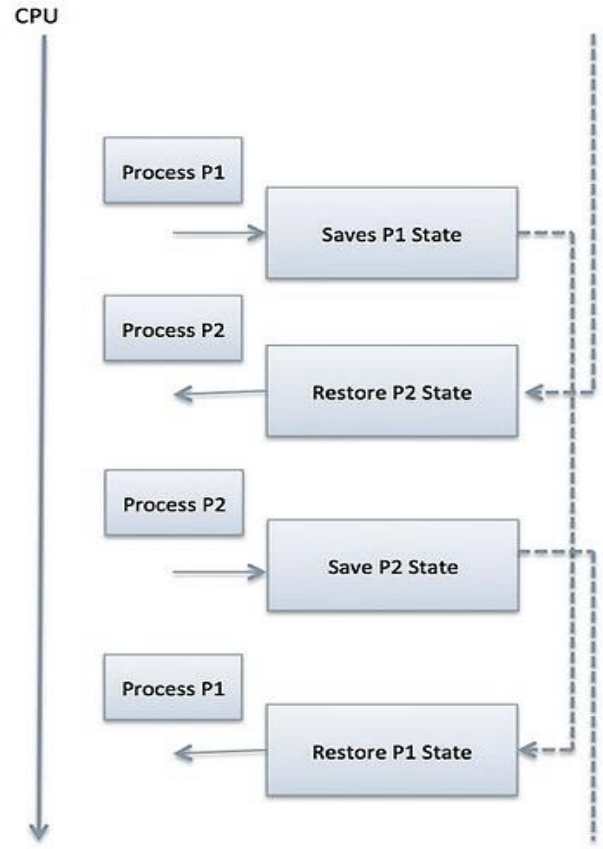


Fig.2.5: कॉन्टेक्स्ट स्विच (Context Switch)

रेजिस्टर आणि मेमरी स्टेट सेव्ह (Save) आणि रिस्टोअर (Restore) करणे आवश्यक असल्याने कॉन्टेक्स्ट स्विचेस संगणकीयदृष्ट्या इंटेन्सिव्ह असतात. कॉन्टेक्स्ट स्विचिंग वेळेचा जास्त वापर टाळण्यासाठी, काही हार्डवेअर सिस्टीम

प्रोसेसर रेजिस्टरचे दोन किंवा अधिक संच वापरतात. जेव्हा प्रोसेस स्विच केली जाते, तेव्हा खालील माहिती नंतरच्या वापरासाठी संग्रहित केली जाते.

- प्रोग्राम काउंटर (Program Counter)
- शेड्यूलिंग माहिती (Scheduling Information)
- बेस आणि लिमिट रेजिस्टर व्हॅल्यू (Value of Base and Limit register)
- सध्या वापरलेले रेजिस्टर (Currently used register)
- बदललेली स्टेट (Changed State)
- I/O स्टेट माहिती (I/O State information)
- अकाउंटिंग माहिती (Accounting information)

2.3 इंटर प्रोसेस कम्युनिकेशन (Inter Process Communication)

ऑपरेटिंग सिस्टममध्ये एकाच वेळी एक्झिक्युट होणाऱ्या प्रोसेस स्वतंत्र (Independent) प्रोसेस किंवा कोऑपरेटिंग (Cooperating) प्रोसेस असू शकतात. जर सिस्टममध्ये एक्झिक्युट होणाऱ्या इतर प्रोसेस मुळे त्यावर परिणाम होत नसेल तर प्रोसेस स्वतंत्र असते. इंटर प्रोसेस कम्युनिकेशन (IPC) ही एक अशी यंत्रणा आहे ज्यामध्ये एका प्रोसेसचा दुसऱ्या प्रोसेसशी कम्युनिकेशन साधला जातो आणि हे सहसा फक्त एकाच सिस्टममध्ये होते. कम्युनिकेशन दोन प्रकारचे असू शकते –

1. फक्त एकाच प्रोसेस पासून सुरू होणाऱ्या संबंधित अनेक प्रोसेस, जसे की प्यारेन्ट (Parent) आणि चाईल्ड (Child) प्रोसेस.
2. असंबंधित (Unreleted) प्रोसेस मधील, किंवा दोन किंवा अधिक वेगवेगळ्या प्रोसेस मधील.

प्रोसेस या दोन मार्गांनी एकमेकांशी कम्युनिकेट करू शकतात:

1. शेअर्ड मेमरी (Shared Memory)
2. मेसेज क्यू (Message Queue)

2.3.1 शेअर्ड मेमरी (Shared Memory)

शेअर्ड मेमरी हा मेमरीचा एक राखीव भाग आहे जो अनेक प्रोसेसला ऍक्सेस करता येतो. हे Fig. 2.5 मध्ये दाखवल्याप्रमाणे शेअर्ड मेमरी भागातील डेटा वाचून आणि लिहून प्रोसेसना एकमेकांशी कम्युनिकेट करण्याची परवानगी देते.

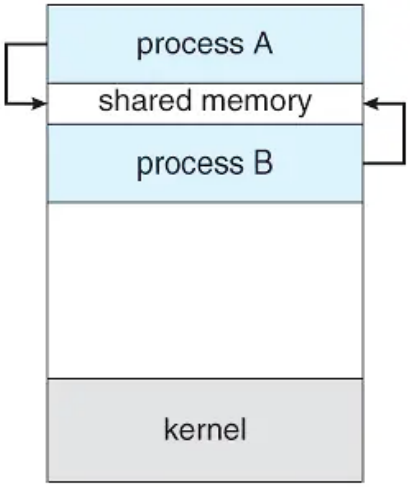


Fig. 2.5: शेअर्ड मेमरी (Shared Memory)

शेअर्ड मेमरी ही प्रोसेसशी कम्युनिकेट करण्याची एक जलद आणि कार्यक्षम मार्ग आहे, परंतु प्रोसेस काळजीपूर्वक सिंक्रोनाइज्ड न केल्यास ती वापरणे कठीण होऊ शकते. शेअर्ड मेमरीचे दोन मुख्य प्रकार आहेत:

1. **अनॉनिमियस शेअर्ड मेमरी (Anonymous shared memory)**: अनॉनिमियस शेअर्ड मेमरी कोणत्याही फाइल किंवा इतर सिस्टम ऑब्जेक्टशी संबंधित नाही. ती ऑपरेटिंग सिस्टमद्वारे तयार केली जाते आणि ती तयार करणाऱ्या प्रोसेससाठीच ती एक्सेसिबल असते.

2. **मॅप्ड शेअर्ड मेमरी (Mapped shared memory)**: मॅप्ड शेअर्ड मेमरी ही फाइल किंवा इतर सिस्टम ऑब्जेक्टशी संबंधित असते. ती एका किंवा अधिक प्रोसेसच्या ॲड्रेस स्पेसमध्ये फाइल मॅप करून तयार केली जाते.

अनेक प्रोसेस एका सामान्य शेअर्ड मेमरीमध्ये प्रवेश करू शकतात. अनेक एक्सेस शेअर्ड मेमरीद्वारे कम्युनिकेट करतात, जिथे एक प्रोसेस एका वेळी बदल करते आणि नंतर इतर प्रोसेस बदल पाहतात. शेअर्ड मेमरी कर्नल (Kernal) वापरत नाही. सर्व POSIX सिस्टीम, तसेच विंडोज ऑपरेटिंग सिस्टीम शेअर्ड मेमरी वापरतात.

2.3.2 मेसेज क्यू (Message Queue)

अनेक प्रोसेस एकमेकांशी कनेक्ट न होता मेसेज क्यूत डेटा वाचू आणि लिहू शकतात. रेसिपीएंट त्यांना परत मिळवेपर्यंत क्यूत संग्रहित केले जातात जे Fig 2.6 मध्ये दर्शविले आहे. मेसेज क्यू इंटर प्रोसेस कॉम्युनिकेशनसाठी खूप उपयुक्त आहेत आणि बहुतेक ऑपरेटिंग सिस्टमद्वारे वापरल्या जातात. जर दोन प्रोसेस p1 आणि p2 एकमेकांशी कम्युनिकेट करू इच्छित असतील, तर त्या खालीलप्रमाणे पुढे जातात:

1. कम्युनिकेशन लिंक स्थापित करणे (जर लिंक आधीच अस्तित्वात असेल, तर ती पुन्हा स्थापित करण्याची आवश्यकता नाही.)
2. बेसिक प्रिमिटिव्हज वापरून मेसेजची देवाणघेवाण सुरू करणे.
3. त्यासाठी आपल्याला किमान दोन प्रिमिटिव्हजची आवश्यकता असते जे खाली दिलेले आहेत.
    a. send(message, destination) or send(message)
    b. receive(message, host) or receive(message)

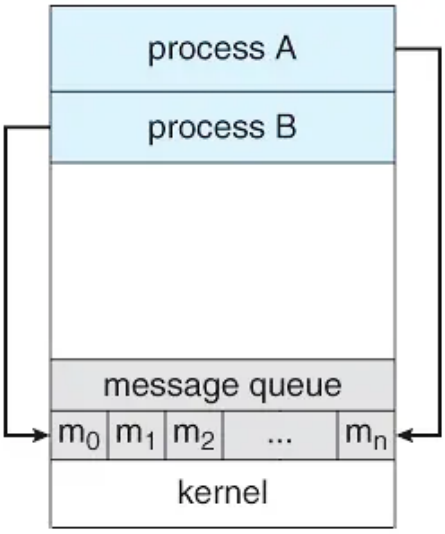


Fig. 2.6: मेसेज क्यू (Message Queue)

2.4 थ्रेड्स (Threads)

थ्रेड म्हणजे प्रोसेस कोडद्वारे एक्झिक्यूशनचा प्रवाह असतो, ज्याचा स्वतःचा प्रोग्राम काउंटर असतो जो पुढे कोणती इंस्ट्रक्शन एक्झिक्युट करायची याचा मागोवा ठेवतो, सिस्टम रजिस्टर्स ज्यामध्ये त्याचे सध्याचे कार्यरत व्हेरिअबल्स असतात आणि स्टॅक (Stack) ज्यामध्ये एक्झिक्यूशन इतिहास असतो. थ्रेड त्याच्या पीअर थ्रेड्ससोबत कोड सेगमेंट, डेटा सेगमेंट आणि ओपन फाइल्स सारखी काही माहिती शेअर करतो. जेव्हा एक थ्रेड कोड सेगमेंट मेमरी आयटम बदलतो तेव्हा इतर सर्व थ्रेड्स ते पाहतात. थ्रेडला लाईट वेट प्रोसेस (Light Weight Process) देखील म्हणतात. थ्रेड्स पॅरेलेलीज्म द्वारे अप्लिकेशन परफॉर्मन्स सुधारण्याचा मार्ग प्रदान करतात. थ्रेड्स हे ओव्हरहेड थ्रेडला क्लासिकल प्रोसेसचे समतुल्य कमी करून ऑपरेटिंग सिस्टमची कार्यक्षमता सुधारण्यासाठी सॉफ्टवेअर दृष्टिकोन दर्शवितात. प्रत्येक थ्रेड अगदी एकाच प्रोसेसचा असतो आणि कोणताही थ्रेड प्रोसेसबाहेर अस्तित्वात असू शकत नाही. प्रत्येक थ्रेड नियंत्रणाचा वेगळा प्रवाह दर्शवितो. नेटवर्क सर्व्हर आणि वेब सर्व्हर अंमलात आणण्यासाठी थ्रेड्स यशस्वीरित्या वापरले गेले आहेत. ते शेअर्ड मेमरी मल्टीप्रोसेसरवर अप्लिकेशनच्या समांतर एक्झिक्युशनसाठी योग्य पाया देखील प्रदान करतात. खालील Fig 2.7 सिंगल-थ्रेडेड आणि मल्टीथ्रेडेड प्रोसेसचे कार्य दर्शवते.

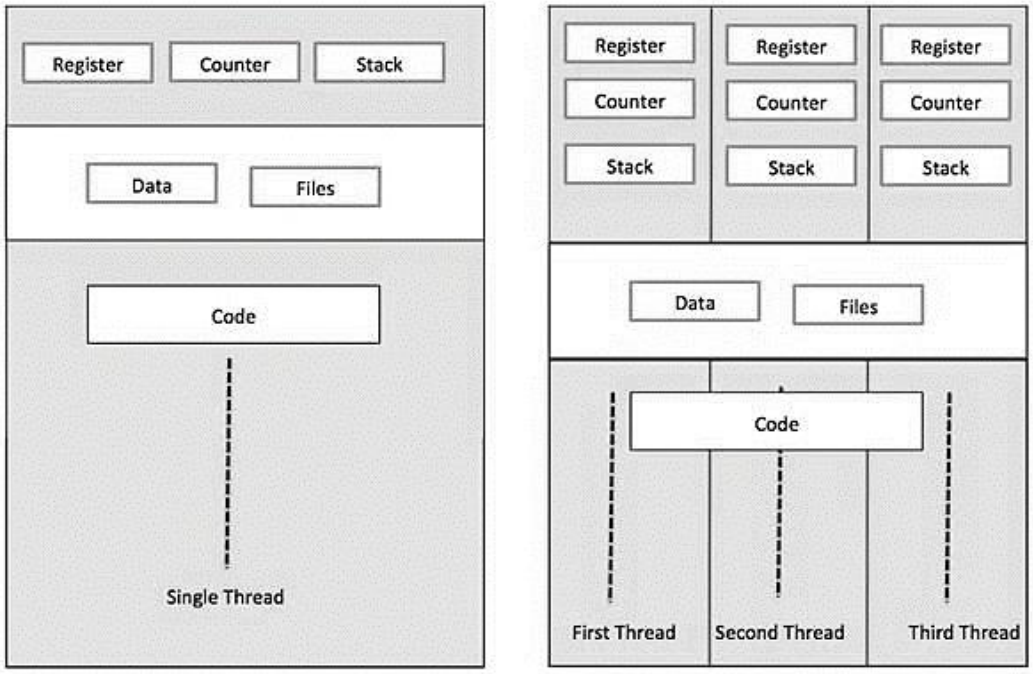


Fig. 2.7: सिंगल-थ्रेडेड आणि मल्टीथ्रेडेड प्रक्रिया प्रोसेस (Single-threaded and a multithreaded process)

2.4.1 थ्रेडचे फायदे (Benefits of Thread)

1. वेगवान कॉन्टेक्स्ट स्विचिंग.
2. कमी मेमरीचा वापर (थ्रेड मेमरी शेअर करतात).
3. सुधारित कामगिरी आणि उत्तम प्रतिसाद.
4. पॅरलल एक्झिक्युशनद्वारे कार्यक्षम सीपीयू वापर.
5. मल्टी-कोर सीपीयू वर अनेक थ्रेड एकाच वेळी रन होऊ शकतात
1. 6. थ्रेड्स मेमरी शेअर केल्यामुळे, इंटर प्रोसेस कम्युनिकेशन (आयपीसी) न वापरता डेटा त्यांच्या दरम्यान सहजपणे शेअर केला जाऊ शकतो.
6. थ्रेड्स अनेक सीपीयू कोरमध्ये वर्कलोड वितरीत करू शकते ज्यामुळे सिस्टमची स्केलेबिलिटी सुधारते.

2.4.2 थ्रेड्सचे प्रकार (Types of Thread)

खालील दोन मार्गांनी थ्रेड इम्प्लिमेंट केले जातात -

a. यूजर लेव्हल थ्रेडस (User Level Threads) - यूजरने व्यवस्थापित थ्रेडस .
b. कर्नल लेव्हल थ्रेड्स (Kernel Level Threads)- ऑपरेटिंग सिस्टम व्यवस्थापित थ्रेड्स कर्नल, ऑपरेटिंग सिस्टम कोअरवर कार्य करतात.

a. यूजर लेव्हल थ्रेडस (User Level Threads)

यात, थ्रेड मॅनेजमेंट कर्नलला थ्रेड्सच्या अस्तित्वाची जाणीव नसते. थ्रेड लायब्ररीमध्ये थ्रेड्स तयार (Create) करण्यासाठी आणि नष्ट (Destroy) करण्यासाठी, थ्रेड्समध्ये संदेश आणि डेटा पास करण्यासाठी, थ्रेड एक्झिक्युशन शेड्यूल करण्यासाठी आणि थ्रेड कॉन्टेक्स्ट सेव्ह आणि रिस्टोअर करण्यासाठी कोड असतो. ॲप्लिकेशन एकाच थ्रेडने सुरू होते. यूजर लेव्हल थ्रेडमध्ये कॉन्टेक्स्ट स्विचिंग वेगवान असते. जर एखादा यूजर लेव्हल थ्रेड ब्लॉकिंग ऑपरेशन करत असेल तर संपूर्ण प्रोसेस ब्लॉक्ड होईल उदा: पोझिक्स थ्रेड्स (POSIX threads), जावा थ्रेड्स (Java threads) इ. यूजर लेवल थ्रेड Fig. 2.8 मध्ये दर्शविले आहे.

फायदे (Advanteges)

1. थ्रेड स्विचिंगला कर्नल मोड विशेषाधिकारांची (Priviledge) आवश्यकता नसते.
2. यूजर लेवल थ्रेड कोणत्याही ऑपरेटिंग सिस्टमवर रन होऊ शकतो.

3. शेड्यूलिंग यूजर लेवल थ्रेड मध्ये विशिष्ट अप्लिकेशन असू शकते.
4. यूजर लेवल थ्रेड तयार (Create) आणि व्यवस्थापित (Manage) करण्यासाठी वेगवान आहेत.
5. यूजर लेव्हल थ्रेडमध्ये कॉन्टेक्स्ट स्विचिंग वेगवान असते.

तोटे (Disadvanteges)

1. ठराविक ऑपरेटिंग सिस्टममध्ये, बहुतेक सिस्टम कॉल ब्लॉक करत असतात.
2. मल्टीथ्रेडेड अप्लिकेशन मल्टीप्रोसेसिंगचा फायदा घेऊ शकत नाही.

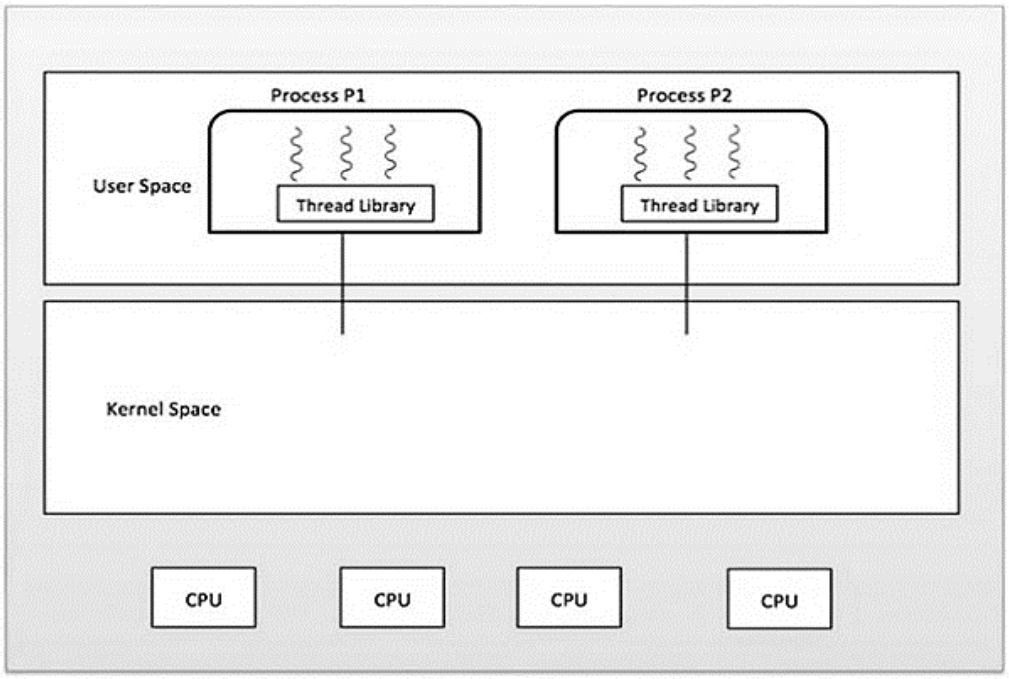


Fig. 2.8: यूजर लेवल थ्रेड (User Level Thread)

b. कर्नल लेव्हल थ्रेड्स (Kernel Level Threads)

यात, थ्रेड मॅनेजमेंट कर्नलद्वारे केले जाते. अप्लिकेशनचा एरियात थ्रेड मॅनेजमेंट कोड नसते. कर्नल थ्रेड्स थेट ऑपरेटिंग सिस्टमद्वारे समर्थित असतात. कोणतेही अप्लिकेशन मल्टीथ्रेड करण्यासाठी प्रोग्राम केला जाऊ शकतो. अप्लिकेशन मधील सर्व थ्रेड एकाच प्रोसेसमध्ये समर्थित असतात. कर्नल संपूर्ण प्रोसेससाठी आणि प्रोसेसमधील वैयक्तिक थ्रेड्ससाठी संदर्भ माहिती ठेवते. कर्नलद्वारे शेड्यूलिंग थ्रेड आधारावर केले जाते. कर्नल कर्नल स्पेसमध्ये थ्रेड निर्मिती, वेळापत्रक (Timetable) आणि मॅनेजमेंट (Management) करते. कर्नल थ्रेड सामान्यत: यूजर लेवल थ्रेडपेक्षा तयार (Create) आणि व्यवस्थापित (Manage) करण्यास हळू असतात. कर्नल-लेवल थ्रेड मध्ये कॉन्टेक्स्ट स्विचिंग हळू असते. जरी एक कर्नल-लेवल थ्रेड ब्लॉकिंग ऑपरेशन करत असेल, तरीही त्याचा इतर थ्रेडवर परिणाम होत नाही. उदा: विंडोज (Windows), सोलारिस (Solaris).

फायदे (Advanteges)

1. कर्नल एकाच प्रोसेस मधील अनेक थ्रेड्स एकाच वेळी अनेक प्रोसेसवर शेड्यूल करू शकतो.
2. प्रोसेस मधील एक थ्रेड ब्लॉक्ड केला असल्यास, कर्नल त्याच प्रोसेसचा दुसरा थ्रेड शेड्यूल करू शकतो.
3. कर्नल रूटीन स्वत: मल्टीथ्रेड केले जाऊ शकतात.

तोटे (Disadvantes)

1. कर्नल थ्रेड सामान्यत: यूजर्सच्या थ्रेडपेक्षा तयार आणि व्यवस्थापित करण्यास हळू असतात.
2. त्याच प्रोसेस मध्ये एका थ्रेडमधून दुसऱ्या थ्रेडवर नियंत्रण हस्तांतरित करण्यासाठी कर्नलवर मोड स्विच आवश्यक असते.
3. कर्नल-लेवल थ्रेड मध्ये कॉन्टेक्स्ट स्विचिंग हळू असते.

2.4.3 मल्टीथ्रेडिंग मॉडेल्स (Multithreading Models)

काही ऑपरेटिंग सिस्टम एकत्रित यूजर लेवल थ्रेड आणि कर्नल लेव्हल थ्रेड सुविधा प्रदान करते. या एकत्रित दृष्टिकोनाचे सोलारिस हे एक चांगले उदाहरण आहे. मल्टीथ्रेडिंग सिस्टिममध्ये, समान अप्लिकेशन मधील अनेक थ्रेड्स अनेक प्रोसेसरवर समांतर चालू शकतात आणि ब्लॉकिंग सिस्टम कॉलला संपूर्ण प्रोसेस ब्लॉक करण्याची आवश्यकता नसते. मल्टीथ्रेडिंग मॉडेलचे तीन प्रकार आहेत.

1. अनेक ते अनेक रिलेशनशिप (Many to many relationship)
2. अनेक ते एक रिलेशनशिप (Many to one relationship)
3. एक ते एक रिलेशनशिप (One to one relationship)

1.अनेक ते अनेक रिलेशनशिप (Many to many relationship)

अनेक-ते-अनेक मॉडेल मल्टिप्लेक्समध्ये कितीही यूजर्स थ्रेड समान किंवा कमी संख्येच्या कर्नल थ्रेडवर जोडले जातात. खालील Fig. 2.9 अनेक-ते-अनेक थ्रेडिंग मॉडेल दर्शविते जिथे 6 यूजर लेवल थ्रेड 6 कर्नल लेव्हल थ्रेड्ससह मल्टिप्लेक्सिंग मध्ये आहेत. या मॉडेलमध्ये, डेव्हलपरच्या आवश्यकतेनुसार बरेच यूजर थ्रेड तयार करू शकतात आणि संबंधित कर्नल थ्रेड मल्टीप्रोसेसर मशीनवर समांतर चालवू शकतात. हे मॉडेल कॉंकरेंसीवरील उत्कृष्ट अचूकता प्रदान करते आणि जेव्हा एखादा थ्रेड ब्लॉकिंग सिस्टम कॉल करतो तेव्हा कर्नल एक्झिक्युशन साठी दुसरा थ्रेड शेड्यूल करू शकतो.

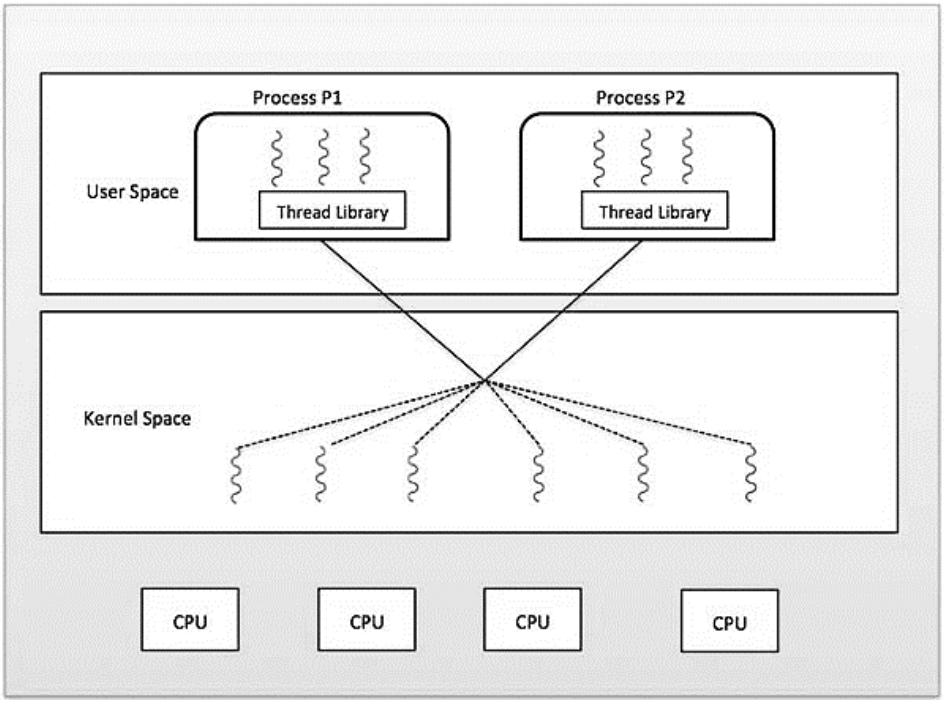


Fig. 2.9: अनेक ते अनेक रिलेशनशिप (Many to many relationship)

2. अनेक ते एक रिलेशनशिप (Many to one relationship)

अनेक ते एक मॉडेल एका कर्नल-लेवल थ्रेडवर बरेच यूजर लेवल थ्रेड्स मॅप करतात. थ्रेड मॅनेजमेंटयूजर्सच्या स्पेसमध्ये थ्रेड लायब्ररीद्वारे केले जाते. जेव्हा थ्रेड ब्लॉकिंग सिस्टम कॉल करते तेव्हा संपूर्ण प्रोसेस ब्लॉक्ड केली जाते. एका वेळी फक्त एकच थ्रेड कर्नल एक्सेस करू शकतो, म्हणून अनेक थ्रेड्स मल्टीप्रोसेसरवर समांतर चालविण्यात अक्षम असतात. जर यूजर लेवल थ्रेड लायब्ररी ऑपरेटिंग सिस्टममध्ये अशा प्रकारे अंमलात आणल्या जाते की सिस्टम त्यांना समर्थन देत नाहीत, तर कर्नल थ्रेड्स अनेक ते एक रिलेशनशिप मोड वापरतात. खालील Fig. 2.10 अनेक-ते- एक थ्रेडिंग मॉडेल दर्शविते.

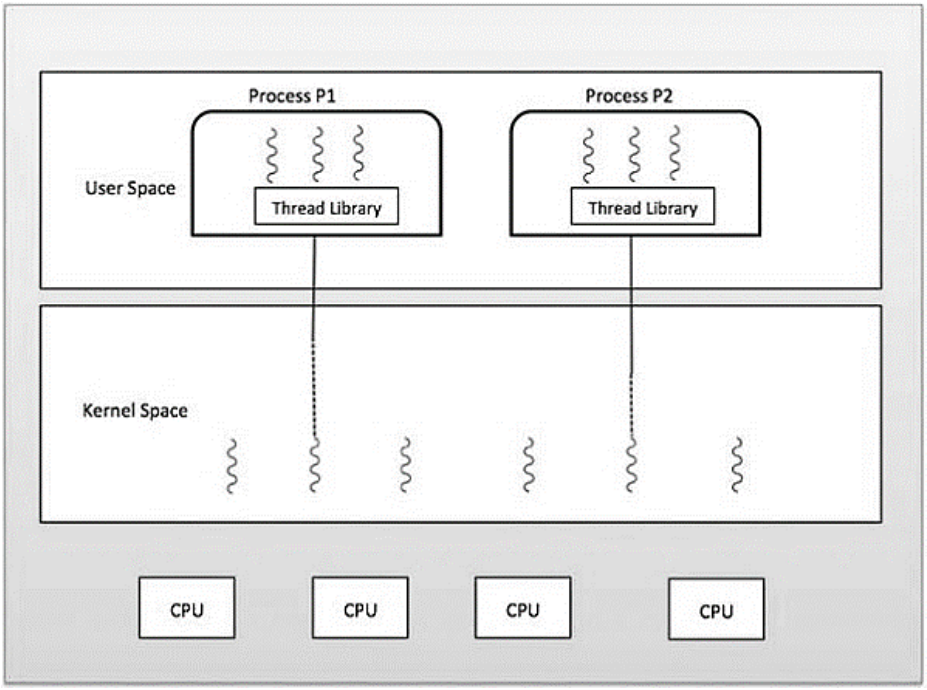


Fig. 2.10: अनेक ते एक रिलेशनशिप (Many to One relationship)

3. एक ते एक रिलेशनशिप (One to one relationship)

यात कर्नल-लेवल थ्रेडवर यूजर लेवल थ्रेडचे एक ते एक संबंध असते. हे मॉडेल अनेक-ते-एक मॉडेलपेक्षा अधिक कॉंकरेंसी (Concurrency) प्रदान करते. जेव्हा एखादा थ्रेड ब्लॉकिंग सिस्टम कॉल करतो तेव्हा हे आणखी एक थ्रेड रन करण्याची अनुमती देते. हे मायक्रोप्रोसेसरवर समांतर एक्झिक्युट करण्यासाठी अनेक थ्रेडचे समर्थन करते. या मॉडेलचा तोटा म्हणजे यूजर लेवल थ्रेड तयार करण्यासाठी संबंधित कर्नल थ्रेड आवश्यक आहे. ओएस/2 (OS/2) , विंडोज एनटी (Windows NT) आणि विंडोज 2000 (Windows 2000) एक ते एक संबंध मॉडेल वापरतात. खालील Fig. 2.11 एक -ते- एक थ्रेडिंग मॉडेल दर्शविते.

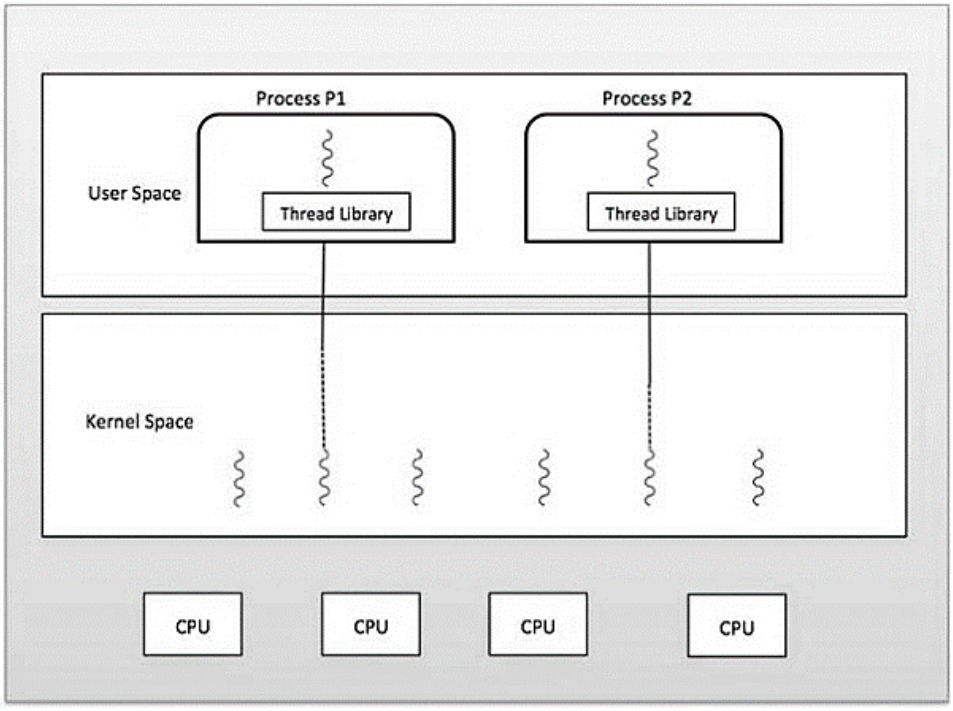


Fig. 2.11: एक ते एक रिलेशनशिप (One to One relationship)

2.5 LINUX मधील प्रोसेस कमांड्स (Process Commands in LINUX)

लिनक्समध्ये प्रोसेस ही एक्झिक्युशनमध्ये असलेल्या प्रोग्रामपेक्षा वेगळी असते आणि एका प्रोग्रामची रनिंग इंस्टन्स असते. आपण एक्झिक्युट केलेली कोणतीही कमांड प्रोसेस सुरू करते. लिनक्समध्ये प्रोसेस दोन प्रकार असू शकतात:

1. **फारग्राउंड प्रोसेस (Forground Process)** : प्रोसेस इनपुटसाठी यूजरवर अवलंबून असते ज्यास इंटरॅक्टिव्ह प्रोसेस म्हणून संबोधले जाते.
2. **बॅकग्राउंड प्रोसेस (Background Process):** प्रोसेस यूजर्सपासून स्वतंत्रपणे रन होते ज्याला नॉन-इंटरॅक्टिव्ह किंवा स्वयंचलित प्रोसेस म्हणतात.

लिनक्स मधील प्रोसेस तयार झाल्यानंतर आणि ती संपुष्टात येण्यापूर्वी वेगवेगळ्या स्टेट्स मधून जाऊ शकते. या स्टेट्स आहेत:

1. **रनिंग (Running)**: रनिंग असलेल्या स्टेटमधील प्रोसेसचा अर्थ असा आहे की ती रन होत आहे किंवा रन होण्यास तयार आहे.
2. **स्लीपिंग (Sleeping)**: प्रोसेस स्लीपिंगच्या स्टेट मध्ये असते जेव्हा ती संसाधन (Resource) उपलब्ध होण्याची प्रतीक्षा करीत असते. इंटरप्टिबल (Interruptible) स्लीपमधील प्रोसेस सिग्नल हाताळण्यासाठी जागे होईल, तर अनइंटरप्टिबल (Uninterruptible) स्लीपमधील प्रोसेस जागे होणार नाही.
3. **स्टॉप्ड (Stopped)**: जेव्हा स्टॉप सिग्नल प्राप्त होतो तेव्हा प्रोसेस स्टॉप्ड स्टेट मध्ये प्रवेश करते.
4. **झोम्बी (Zombie)**: ही अशी स्थिती आहे ज्यामध्ये प्रोसेस मृत असते परंतु प्रोसेसची नोंद अजूनही टेबलमध्ये असते.

लिनक्समध्ये रनिंग मध्ये असलेल्या प्रोसेसचा मागोवा घेण्यासाठी दोन कमांड उपलब्ध आहेत. या दोन कमांड **top** आणि **ps** आहेत.

1. लिनक्स प्रोसेस मॅनेज करण्यासाठी top कमांड (top Command for Managing Linux Processes)

top (टेबल ऑफ प्रोसेसेस-Table of processes) कमांड लिनक्समध्ये रनिंग असलेल्या प्रोसेसचे डायनॅमिक, रिअल-टाइम विव्ह (Real Time View) आणि कर्नल-मॅनेज्ड कार्ये दर्शविते. कमांड सीपीयू आणि मेमरी वापरासह संसाधनाचा उपयोग दर्शविणारी सिस्टम माहितीचे सारांश (Summary) देखील प्रदान करते. आपल्या मशीनवरील रन होत असलेल्या प्रोसेसचा मागोवा घेण्यासाठी आपण top कमांड वापरू शकता. top कमांड Fig. 2.12 मध्ये दाखवल्याप्रमाणे रिअल-टाइममध्ये चालू असलेल्या प्रोसेसेसची यादी त्यांच्या मेमरी आणि सीपीयू वापरासह प्रदर्शित करते.

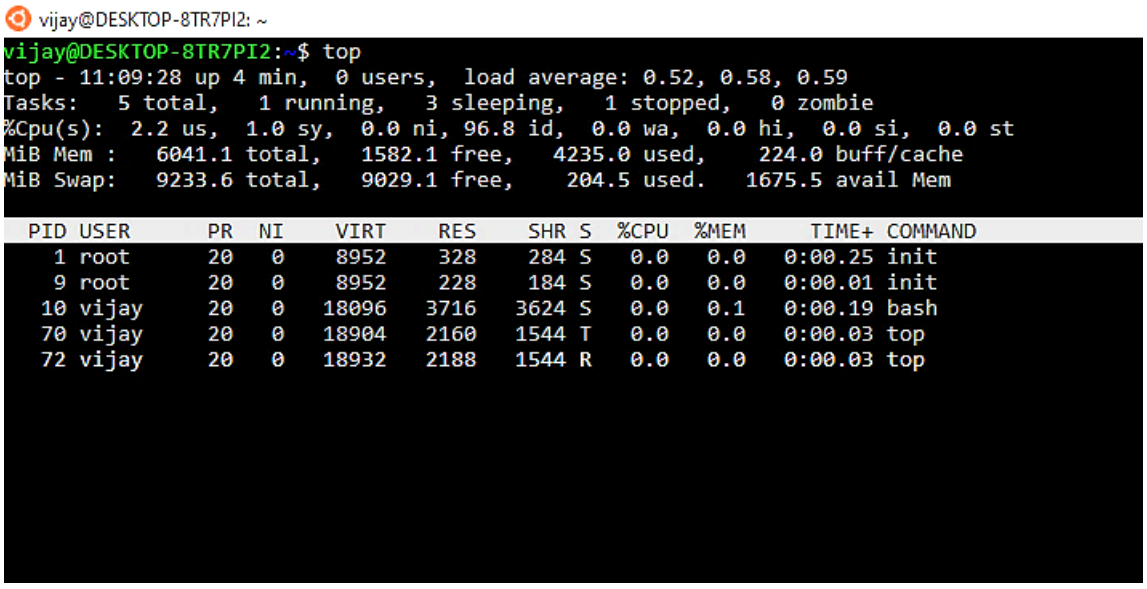


Fig. 2.12: टॉप कमांडद्वारे रन होत असलेल्या प्रोसेसचे रिअल-टाइम दृश्य (Real-time view of running processes by top command)

top कमांडचा आउटपुट मध्ये खालील माहिती असते.

1. **PID**: प्रत्येक प्रोसेसला दिलेला युनिक प्रोसेस आयडी (Unique Process ID given to each process)

2. **User**: प्रोसेसच्या मालक असलेल्या यूजर्सचे नाव (Username of the process owner)
3. **PR**: शेड्यूलिंग करताना प्रोसेसची प्रायोरिटी (Priority given to a process while scheduling)
4. **NI:** प्रोसेसची 'नाईस ' व्हॅलू ('nice' value of a process)
5. **VIRT**: प्रोसेसद्वारे वापरल्या जाणाऱ्या व्हर्च्युअल मेमरीचे प्रमाण (Amount of virtual memory used by a process)
6. **RES**: प्रोसेसद्वारे वापरल्या जाणाऱ्या फिजिकल मेमरीचे प्रमाण (Amount of physical memory used by a process)
7. **SHR**: इतर प्रोसेससह शेअर केलेल्या मेमरीचे प्रमाण (Amount of memory shared with other processes)
8. **S**: प्रोसेसची स्टेट (State of the process)
    1. 'D' = अनइंटरप्टिबल स्लीप (uninterruptible sleep)
    2. 'R' = रनिंग (running)
    3. 'S' = स्लीपिंग (sleeping)
    4. 'T' = स्टॉप्ड **(**stopped)
    5. 'Z' = झोम्बी (zombie)
9. **%CPU**: प्रोसेसद्वारे वापरल्या जाणाऱ्या सीपीयूची टक्केवारी (Percentage of CPU used by the process)
10. **%MEM**; प्रोसेसद्वारे वापरल्या जाणाऱ्या रॅमची टक्केवारी (Percentage of RAM used by the process)
11. **TIME+:** प्रोसेसद्वारे वापरलेले एकूण सीपीयूचा वेळ (Total CPU time consumed by the process)
12. **Command**: प्रोसेस सक्रिय करण्यासाठी वापरली जाणारी कमांड (Command used to activate the process)

2. ps कमांड (command)

ps कमांड 'प्रोसेस स्टेटस' चे संक्षिप्त शब्द हे. सध्या रनिंग असलेल्या प्रक्रिया Fig. 2.13 मध्ये दाखविले आहे. तथापि**, top** कमांडच्या विपरीत, जनरेट केलेले आउटपुट रिअल टाइममध्ये नसते.

```
vijay@DESKTOP-8TR7PI2:~$ ps
  PID TTY          TIME CMD
   10 tty1     00:00:00 bash
   70 tty1     00:00:00 top
   78 tty1     00:00:00 top
   84 tty1     00:00:00 ps
vijay@DESKTOP-8TR7PI2:~$
```

Fig.1.13: ps कमांडचे आउटपुट (Output of ps command)

टर्मिनॉलॉजि खालीलप्रमाणे आहे:

PID - प्रोसेस आयडी (process ID)

TTY - टर्मिनलचा टाईप (terminal type)

TIME - प्रोसेस रन झाल्याचा एकूण वेळ (total time the process has been running)

CMD - प्रोसेस सुरू करणाऱ्या कमांडचे नाव (name of the command that launches the process)

अधिक माहिती मिळविण्यासाठी, **ps** कमांड ऑपशन **-u** सहित Fig. 1.14 मध्ये दर्शविल्याप्रमाणे वापरा

```
vijay@DESKTOP-8TR7PI2:~$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
vijay       10  0.0  0.0  18096  3704 tty1     S    11:05   0:00 -bash
vijay       70  0.0  0.0  18904  2000 tty1     T    11:05   0:00 top
vijay       78  0.0  0.0  18932  2012 tty1     T    12:47   0:00 top
vijay       85  0.0  0.0  18648  1920 tty1     R    15:48   0:00 ps -u
vijay@DESKTOP-8TR7PI2:~$
```

Fig.1.14: ps -u कमांडचे आउटपुट (Output of ps –u command)

येथे:

%CPU - प्रोसेस घेत असलेल्या काम्पुटिंग पॉवरचे (Computing Power) प्रमाण दर्शवते.

%MEM - प्रोसेस वापर करत असलेल्या मेमरीचे प्रमाण दर्शवते.

STAT - प्रोसेसचे स्टेट दर्शवते

ps कमांड फक्त सध्या रनिंग असलेल्या प्रोसेस दाखवते, तर तुम्ही Fig 1.15 मध्ये दाखवल्याप्रमाणे पर्याय **–A** वापरून सर्व प्रोसेसची लिस्ट करण्यासाठी देखील त्याचा वापर करू शकता.

```
vijay@DESKTOP-8TR7PI2:~$ ps -A
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    9 tty1     00:00:00 init
   10 tty1     00:00:00 bash
   70 tty1     00:00:00 top
   78 tty1     00:00:00 top
   91 tty1     00:00:00 ps
vijay@DESKTOP-8TR7PI2:~$
```

Fig.1.15: ps -A कमांडचे आउटपुट (Output of ps –A command)

संपूर्ण माहिती मिळविण्यासाठी ps कमांड तुम्ही Fig 1.16 मध्ये दाखवल्याप्रमाणे -f ऑपशन सोबत वापरा .

```
vijay@DESKTOP-8TR7PI2:~$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
vijay         10     9  0 11:05 tty1     00:00:00 -bash
vijay         70    10  0 11:05 tty1     00:00:00 top
vijay         78    10  0 12:47 tty1     00:00:00 top
vijay         92    10  0 15:53 tty1     00:00:00 ps -f
vijay@DESKTOP-8TR7PI2:~$
```

Fig.1.16: ps -f कमांडचे आउटपुट (Output of ps –f command)

3. प्रोसेस थांबविण्यासाठी kill कमांड (kill command to stop a process)

लिनक्समध्ये प्रोसेस थांबविण्यासाठी, 'kill' कमांड वापरा. किल कमांड प्रोसेसला सिग्नल पाठवते. आपण पाठवू शकता असे विविध प्रकारचे सिग्नल आहेत. तथापि, सर्वात सामान्य म्हणजे 'kill -9' जे 'SIGKILL' आहे. Fig 1.17 मध्ये दर्शविल्यानुसार आपण L पर्याय वापरुन सर्व सिग्नल सूचीबद्ध करू शकता.

```
vijay@DESKTOP-8TR7PI2:~$ kill -L
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
vijay@DESKTOP-8TR7PI2:~$
```

Fig.1.17: kill -L कमांडचे आउटपुट (Output of kill -L command)

kill कमांडचे सिंट्याक्स **(Syntax)**: **$ kill [signal] PID**

4. wait कमांड (command)

wait कमांड ही एक शेल कमांड आहे जी बॅकग्राउंड रनिंग प्रोसेस पूर्ण होण्याची वाट पाहते आणि एक्झिट स्टेटस परत करते. बॅश स्क्रिप्टमध्ये प्रतीक्षा करताना तीन अतिरिक्त पॅरामीटर्स आहेत:

1. कमांड नंतर असलेले अँपरसँड चिन्ह (**&**) बॅकग्राउंड जॉब दर्शवते.
2. **$!** शेवटच्या बॅकग्राउंड प्रोसेसचा PID मिळवते. अनेक बॅकग्राउंड प्रोसेससह काम करताना मागील PID एका व्हेरिएबलमध्ये साठवते.

3. **$?** शेवटच्या प्रोसेसची एक्झिट स्टेटस प्रिंट करते.

हे तीन पॅरामीटर्स एकत्र कसे काम करतात हे पाहण्यासाठी, टर्मिनल विंडो उघडा आणि Fig. 1.18 मध्ये दाखवल्याप्रमाणे रन करा.

```
vijay@DESKTOP-8TR7PI2:~$ sleep 30 &
[3] 161
vijay@DESKTOP-8TR7PI2:~$ wait $!
[3]   Done                    sleep 30
vijay@DESKTOP-8TR7PI2:~$ wait $?
-bash: wait: pid 0 is not a child of this shell
vijay@DESKTOP-8TR7PI2:~$
```

Fig. 1.18: wait कमांडमध्ये वापरल्या जाणाऱ्या अतिरिक्त पॅरामीटर्सचे आउटपुट (Output of additional parameters used in wait command)

5. sleep कमांड (command)

डमी जॉब तयार करण्यासाठी स्लीप कमांड वापरला जातो. डमी जॉब एक्झिक्युशनला उशीर करण्यास मदत करतो. हि कमांड वेळ डीफॉल्टनुसार सेकंदात घेतो, परंतु शेवटी एक छोटा सफीक्स (s, m, h, d) लावून ते इतर कोणत्याही फॉरमॅटमध्ये रूपांतरित करता येते. ही कमांड NUMBER द्वारे परिभाषित केलेल्या वेळेसाठी एक्झिक्युशनला विराम देते. लिनक्समधील `स्लीप` कमांडची सिंट्याक्स **(Syntax)** खाली दिली आहे.

sleep NUMBER[SUFFIX]...

जिथे,

NUMBER हा कमांड किती काळासाठी स्लीप करायचा आहे ते दर्शवतो.

SUFFIX चा वापर वेळेचे एकक निर्दिष्ट करण्यासाठी केला जाऊ शकतो (सेकंदांसाठी s, मिनिटांसाठी m, तासांसाठी h, इ.).

उदाहरण: $sleep 20

$sleep 20m

$sleep 1h

6. exit कमांड (command)

लिनक्समधील एक्झिट कमांड ही एक बिल्ट-इन कमांड आहे, जी सध्या जिथे एक्झिक्युट होत आहे तिथे शेल बंद करण्यासाठी वापरली जाते. लिनक्समधील एक्झिट कमांडची मूलभूत सिंट्याक्स (Syntax) खालीलप्रमाणे आहे:

exit [n]

जिथे n हा पर्यायी आर्ग्युमेंट आहे तो एक्झिट स्टेटस दर्शवतो.

लिनक्समधील एक्झिट कमांडला कोणताही पर्याय नसतो, ही कमांड कोणत्याही एक्झिट स्टेटस परत न करता फक्त चालू शेल बंद करते. लिनक्समध्ये एक्झिट कमांडचा वापर करून, ही कमांड शेल बंद करते आणि एक्झिट स्टेटस अर्ग्युमेंट नंबरसह परत करते.

7. nice कमांड (command)

"नाइस व्हॅल्यू **(Nice Value)**" म्हणून ओळखल्या जाणाऱ्या विशिष्ट प्रायोरिटीसह नवीन प्रोसेस सुरू करण्यासाठी ही कमांड वापरली जाते. जास्त नाइस व्हॅल्यू प्रोसेसची प्रायोरिटी कमी करते, तर कमी (निगेटिव्ह) नाइस व्हॅल्यू ती वाढवते. जास्त प्रायोरिटी असलेल्या प्रोसेसना अधिक CPU वेळ मिळतो.

References:

1. Operating System: A Concept-Based Approach by Dhananjay M. Dhamdhere, McGraw Hill Education 3rd edition, ISBN: 978-1259005589
2. Operating Systems : Internals and Design Principles by William Stallings, Pearson Education 9th Edition, ISBN: 978-9352866717

3. Linux The Complete Reference by Richard Petersen, McGraw Hill, 6th edition, ISBN: 978-0071492478
4. Linux command line and shell scripting by Richard Blum, Wiley India, ISBN: 978-1118983843
5. Operating System Concepts by Abraham Silberschatz and James Peterson, Wiley India, ISBN: 9781119454083

Websites:

1. https://www.geeksforgeeks.org/ipc-shared-memory/
2. https://www.javatpoint.com/what-is-inter-process-communication
3. https://www.javatpoint.com/system-calls-in-operating-system
4. https://www.linux.org/
5. https://www.tutorialspoint.com/operating_system/os_linux.htm
6. https://opensource.com/resources/linux
7. https://www.geeksforgeeks.org/inter-process-communication-ipc/
8. https://www.geeksforgeeks.org/introduction-of-system-call/
9. https://www.tutorialspoint.com/what-is-interprocess-communication
10. https://www.scaler.com/topics/operating-system/inter-process-communication-in-os/
11. https://www.tutorialspoint.com/what-is-process-control-block-pcb
12. https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/
13. https://www.javatpoint.com/os-process-states
14. https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/
15. https://www.geeksforgeeks.org/thread-in-operating-system/
16. https://www.javatpoint.com/threads-in-operating-system

E-learning material (Video lectures)

1. https://www.youtube.com/watch?v=LOfGJcVnvAk – Threads in OS
2. https://www.techtarget.com/whatis/definition/multithreading
3. https://www.youtube.com/watch?v=fxMOi7BlTyM - Threads in OS
4. https://www.youtube.com/watch?v=jZ_6PXoaoxo – For Process State
5. https://www.youtube.com/watch?v=GuGjNaMl0Rs – For PCB
6. https://www.youtube.com/watch?v=hWR8fOtD_g4 –For PCB
7. https://www.youtube.com/watch?v=iLC--q7O_KM _ Process Scheduling Queue
8. https://www.youtube.com/watch?v=dJuYKfR8vec - For IPC
9. https://www.youtube.com/watch?v=fSMVWmGPqlM - For IPC
10. https://www.youtube.com/watch?v=_9P1d-si4b4 - For IPC

युनिट – 3
सीपीयू शेड्यूलिंग
(CPU Scheduling)

विषय निष्पत्ती (Course Outcome):

CO3: विविध CPU शेड्युलिंग अल्गोरिथम इम्प्लिमेंट करा आणि त्यांच्या प्रभावीतेचे मूल्यांकन करा.

घटक निष्पत्ती (Theory Learning Outcome -TLO):

1. दिलेल्या शेड्यूलिंग निकषांची आवश्यकता संबंधित उदाहरणासह स्पष्ट करा..
2. दिलेल्या प्रोसेसला CPU वाटप करण्याची प्रक्रिया उदाहरणासह स्पष्ट करा.
3. दिलेल्या शेड्यूलिंग अल्गोरिथमचा टर्नअराउंड टाइम आणि सरासरी वेटिंग टाइम काढा..
4. डेडलॉकला कारणीभूत ठरणाऱ्या दिलेल्या आवश्यक परिस्थितींचे कार्य स्पष्ट करा.

3.1 शेड्यूलिंग (Scheduling)

प्रोसेस शेड्यूलिंग ही प्रोसेस मॅनेजरची क्रिया आहे जी CPU मधून रनिंग प्रोसेस काढून टाकणे आणि विशिष्ट धोरणावर आधारित दुसरी प्रोसेस निवडणे हाताळते. प्रोसेस शेड्यूलिंग हे मल्टीप्रोग्रामिंग ऑपरेटिंग सिस्टमचा एक आवश्यक भाग आहे. अशा ऑपरेटिंग सिस्टम एका वेळी एक्झिक्युटेबल मेमरीमध्ये एकापेक्षा जास्त प्रोसेस लोड करण्यास अनुमती देतात आणि लोड केलेली प्रोसेस टाइम मल्टिप्लेक्सिंग (Time multiplexing) वापरून CPU शेअर करते.

3.1.1 शेड्यूलिंग संकल्पना (Scheduling Concept)

ऑपरेटिंग सिस्टीममध्ये, शेड्युलिंग म्हणजे CPU वेगवेगळ्या प्रोसेससाठी वेळ कसा वाटतो हे व्यवस्थापित करण्याची प्रक्रिया आहे. पुढील कोणती प्रोसेस किती काळ रन होईल हे ऑपरेटिंग सिस्टिम ठरविते, ज्यामुळे अनेक प्रोसेस एकाच वेळी रन होऊ शकतात याची खात्री होते. कार्यक्षम संसाधन वापर (Efficient resource utilization), प्रतिसाद (Responsiveness) आणि एकूण सिस्टम कामगिरीसाठी प्रभावी शेड्यूलिंग अत्यंत महत्त्वाचे आहे. प्रोसेस शेड्यूलिंगची उद्दिष्टे खाली दिली आहेत.

1. **कार्यक्षम संसाधनाचा उपयोग (Efficient Resource Utilization)**: शेड्यूलिंग हे सुनिश्चित करते की सीपीयू व्यस्त ठेवला आहे, निष्क्रिय वेळ रोखत आहे आणि संसाधनाचा उपयोग जास्तीत जास्त करतो.
2. **सुधारित यूजरचा अनुभव (Improved User Experience)**: प्रभावी शेड्यूलिंग एकापेक्षा जास्त प्रोसेस एकाच वेळी रन करण्यास अनुमती देते, रिस्पॉन्सिव्ह आणि इंटरॅक्टिव्ह वातावरण प्रदान करते.
3. **सिस्टम स्थिरता (System Stability)**: शेड्यूलिंग अल्गोरिथम CPU वेळ व्यवस्थापित करण्यास मदत करते आणि प्रोसेसना CPU वर मक्तेदारी करण्यापासून रोखते, ज्यामुळे सिस्टम स्थिरता सुनिश्चित होते..
4. **निष्पक्षता (Fairness)** : शेड्यूलिंगचे उद्दीष्ट सर्व प्रोसेसला सीपीयू वेळेचा योग्य वाटा प्रदान करणे आहे, ज्यामुळे कोणत्याही प्रोसेसला वर्चस्व निर्माण होण्यापासून प्रतिबंधित करते.

3.1.2 सीपीयू आणि आय/ओ बर्स्ट सायकल (CPU and I/O Burst Cycle)

सीपीयू आणि आय/ओ बर्स्ट सायकल ही ऑपरेटिंग सिस्टममध्ये एक मूलभूत संकल्पना आहे, जी Fig.3.1 मध्ये दर्शविल्यानुसार सीपीयू एक्झिक्यूशन आणि आय/ओ वेट स्टेट मध्ये कसे स्विच करते याचे वर्णन करते. प्रक्रिया सामान्यतः CPU बर्स्ट (कॅल्क्युलेशन करणे) ने सुरू होते, त्यानंतर I/O बर्स्ट (I/O ऑपरेशन्स पूर्ण होण्याची वाट पाहणे) येते आणि प्रोसेस पूर्ण होईपर्यंत ही सायकल पुनरावृत्ती होते. शेवटचा CPU बर्स्ट दुसऱ्या I/O बर्स्टऐवजी एक्झिक्युशन समाप्त करण्यासाठी सिस्टम विनंतीसह समाप्त करते. प्रोसेसच्या एक्झिक्युशन दरम्यान हे सायकल अनेक वेळा पुनरावृत्ती होते जोपर्यंत प्रोसेस संपुष्टात येत नाही. प्रोसेस सतत सीपीयूमध्ये एक्झिक्युट होत नाही. त्याऐवजी, ते दोन टप्प्यांमध्ये बदलते:

1. **सीपीयू बर्स्ट (CPU Burst)**: ही प्रोसेस सीपीयूमध्ये एक्झिक्युट होते, अरीथम्याटिक, लॉजिकल ऑपरेशन्स किंवा इंस्ट्रक्शन एक्झिक्युशन सारखे कंप्युटेशन करते.
2. **आय/ओ बर्स्ट (I/O Burst)**: ही प्रोसेस आय/ओ ऑपरेशनची वाट पाहते, जसे की डिस्कवरून वाचन (Reading from disk), नेटवर्क कम्युनिकेशन किंवा यूजर इनपुट.

I/O-बाउंड प्रोसेसमध्ये **(**I/O-bound process) सामान्यतः अनेक लहान CPU बर्स्ट असतात आणि CPU-बाउंड प्रोसेस **(**CPU-bound process) मध्ये काही खूप लांब CPU बर्स्ट असू शकतात.

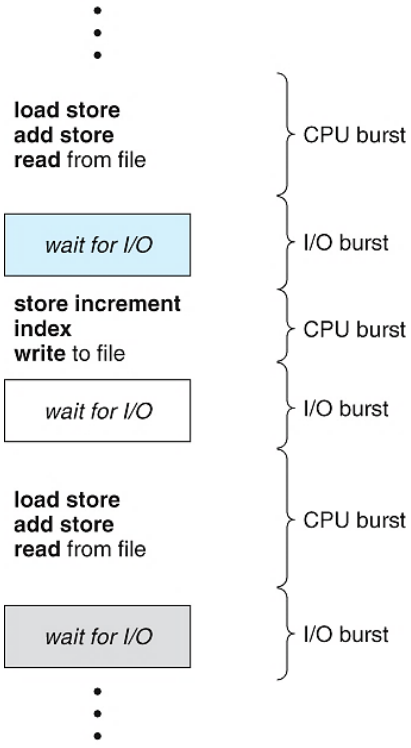


Fig. 3.1: सीपीयू आणि आय/ओ बर्स्ट सायकल (CPU and I/O Burst Cycle)

3.2 शेड्यूलिंगचे प्रकार (Scheduling Types)

खाली दिलेल्या दोन प्रकारच्या शेड्यूलिंग पद्धती आहेत.

3.2.1 प्री- एम्प्टिव्ह शेड्यूलिंग (Pre-emptive Scheduling)

जेव्हा एखादी प्रोसेस रनिंग स्टेट **(**Running state) मधून रेडी स्टेट **(**Ready state) किंवा वेटिंग स्टेट **(**Waiting state) मधून रेडी स्टेट **(**Ready state) मध्ये स्विच करते तेव्हा प्री- एम्प्टिव्ह शेड्यूलिंग वापरली जाते. संसाधने (प्रामुख्याने सीपीयू सायकल) मर्यादित वेळेसाठी प्रोसेसला वाटप केली जातात आणि नंतर ती काढून टाकली जाते आणि जर त्या प्रोसेसला अद्याप CPU बर्स्टचा वेळ शिल्लक असेल तर ही प्रक्रिया पुन्हा रेडी क्यूत ठेवली जाते. ती प्रोसेस एक्झिक्युट करण्याची पुढील संधी येईपर्यंत रेडी क्यूत **(**Queue) राहते.

3.2.2 नॉन-प्री-एम्प्टिव्ह शेड्युलिंग (Non-Pre-emptive Scheduling)

जेव्हा प्रोसेस टर्मिनेट होते किंवा प्रोसेस रेडी स्टेट मधून वेटिंग स्टेटमध्ये जाते तेव्हा नॉन-प्री-एम्प्टिव्ह शेड्युलिंग वापरले जाते. या शेड्युलिंगमध्ये, एकदा संसाधने (सीपीयू सायकल) प्रोसेसला वाटप झाल्यानंतर, प्रक्रिया संपुष्टात येईपर्यंत किंवा वेटिंग स्टेटमध्ये येईपर्यंत प्रोसेस सीपीयूचा ताबा ठेवते. नॉन-प्री-एम्प्टिव्ह शेड्यूलिंगच्या बाबतीत एक्झिक्युशनच्या मध्यभागी सीपीयू वापरण्याच्या, प्रोसेसला व्यत्यय आणत नाही. त्याऐवजी, प्रोसेस त्याचा CPU बर्स्ट वेळ पूर्ण होईपर्यंत वाट पाहते आणि नंतर ते CPU ला दुसऱ्या प्रोसेसला वाटप करू शकते.

3.2.3 शेड्यूलिंगचे निकष (Scheduling criteria)

विशिष्ट परिस्थिती आणि वातावरणासाठी "सर्वोत्तम" शेड्युलिंग अल्गोरिथम **(**Algorithm) निवडण्याचा प्रयत्न करताना विचारात घेण्यासाठी अनेक वेगवेगळे निकष आहेत, ज्यात हे समाविष्ट आहे:

1. **सीपीयू वापर (CPU utilization):** आदर्शपणे सीपीयू 100**%** वेळ व्यस्त असायला पाहिजे, जेणेकरून शून्य सीपीयू सायकल वाया जाऊ नयेत. वास्तविक सिस्टममध्ये सीपीयू वापर 40**%** (हलके लोड केलेले) ते 90**%** (जास्त लोड केलेले) पर्यंत असावा.

2. **थ्रूपुट (Throughput):** प्रति युनिट वेळ पूर्ण केलेल्या प्रोसेसची संख्या. विशिष्ट प्रक्रियेवर अवलंबून 10 / सेकंद ते 1 / तास असू शकते.
3. **टर्नअराऊंड टाइम (Turnaround time)** - सबमिशन वेळेपासून ते पूर्ण होईपर्यंत विशिष्ट प्रोसेस पूर्ण होण्यासाठी लागणारा वेळ.
4. **वेटिंग टाइम (Waiting time):** सीपीयूवर येण्यासाठी प्रोसेस रेडी क्यूत किती वेळ घालवतात
5. **रिस्पॉन्स टाइम (Response time):** कमांड जारी केल्यापासून त्या कमांडला प्रतिसाद देण्यास सुरुवात होईपर्यंत इंटरॅक्टिव्ह प्रोग्रामला लागणारा वेळ.

सीपीयू शेड्यूलिंगमध्ये वापरल्या जाणाऱ्या शब्दावली **(Terminologies Used in CPU Scheduling)**

1. **अराईवल टाइम (Arrival Time)**: ज्या वेळेस प्रोसेस रेडी क्यूत येते.
2. **कंप्लिशन टाइम (Completion Time)** :ज्या वेळेस प्रोसेस त्याचे एक्झिक्युशन पूर्ण करते.
3. **बर्स्ट टाइम (Burst Time)**: सीपीयू एक्झिक्युशनसाठी प्रोसेसला लागणार आवश्यक वेळ.
4. **टर्नअराऊंड टाइम (Turn Around Time)** : कंप्लिशन टाइम आणि अराईवल टाइम दरम्यानचा टाईमचा फरक.

   टर्नअराऊंड टाइम = कंप्लिशन टाइम - अराईवल टाइम
   (Turn Around Time = Completion Time – Arrival Time)
5. **वेटिंग टाइम (Waiting Time)**: टर्नअराऊंड टाइम आणि बर्स्ट टाइम दरम्यानचा टाईमचा फरक.

   वेटिंग टाइम = टर्नअराऊंड टाइम - बर्स्ट टाइम
   (Waiting Time = Turn Around Time – Burst Time)

3.3 शेड्यूलिंग अल्गोरिथमचे प्रकार (Types of Scheduling Algorithm)

ऑपरेटिंग सिस्टममध्ये शेड्यूलिंग अल्गोरिथम सीपीयूवर प्रोसेस कशा एक्झिक्युट केल्या जातात हे व्यवस्थापित करतात. संसाधनाचा वापर ऑप्टोमाईज (Optimize) करणे, वेटिंग टाइम कमी करणे आणि निष्पक्षता सुनिश्चित करणे हे त्यांचे लक्ष्य आहे. हे अल्गोरिथम एकतर प्री-एम्प्टिव्ह किंवा नॉन-प्रीएम्प्टिव्ह अल्गोरिथम असू शकतात आणि डिझाइन केले असतात जेणेकरून एकदा प्रोसेस रनिंग असलेल्या स्टेटमध्ये प्रवेश झाल्यावर; जोपर्यंत तो आपला वाटप केलेला वेळ पूर्ण करेपर्यंत हे प्रीएम्प्ट केले जाऊ शकत नाही, तर प्रीएम्प्टिव्ह शेड्यूलिंग प्राधान्यानुसार असते जेथे हाय प्रायोरिटी प्रोसेस रेडी स्टेटमध्ये प्रवेश केल्यावर शेड्यूलर कमी प्रायोरिटी रनिंग प्रोसेसला प्रीएम्प्ट केले जाऊ शकते.

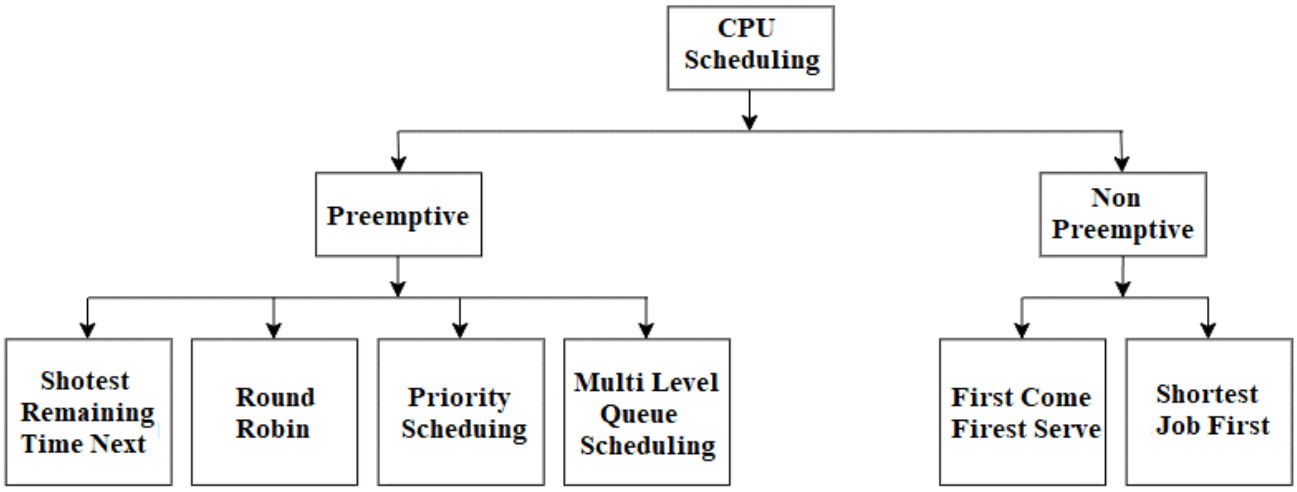


Fig. 3.2: शेड्यूलिंग अल्गोरिथमचे प्रकार (Types of Scheduling Algorithm)

3.3.1 फर्स्ट कम फर्स्ट सर्व्ह (First Come First Serve -FCFS)

फर्स्ट कम फर्स्ट सर्व्ह शेड्यूलिंगमध्ये, रेडी क्यूत प्रथम येणारी प्रोसेसला प्रथम सीपीयू नियुक्त केली जाते. टायच्या बाबतीत म्हणजेच जर दोन किंवा अनेक प्रोसेस एकाच वेळेस आल्यास, लहान प्रोसेस आयडीसह प्रोसेस प्रथम एक्झिक्युट केली जाते. हे नेहमीच नॉन- प्रीएम्प्टिव्ह असते. फर्स्ट कम फर्स्ट सर्व्ह आधारावर जॉब्स एक्झिक्युट केले जातात. हे एक नॉन प्री-प्रीएम्प्टिव्ह अल्गोरिथम आहे त्यामुळे समजणे आणि इम्प्लिमेंट करणे सोपे असते. फर्स्ट कम फर्स्ट सर्व्ह शेड्यूलिंगचे

इम्प्लिमेंटेशन फिफो क्यूवर (FIFO queue) आधारित आहे. सरासरी वेटिंग टाइम जास्त असल्याने कामगिरीमध्ये खराब असते.

फायदे (Advantages)

1. सोपे आणि समजण्यास सोपे आहे.
2. क्यू डेटा स्ट्रक्चर वापरून ते सहजपणे अंमलात आणता येते.
3. स्टार्व्हेशन (Starvation)होत नाही.
4. फर्स्ट कम फर्स्ट सर्व्ह मध्ये शेड्यूलिंग ओव्हरहेड कमी आहे कारण त्यात वारंवार कॉनटेक्स्ट स्विच (Context switches) किंवा जटिल शेड्यूलिंग निर्णयांचा समावेश नाही.
5. FCFS दीर्घकाळ चालणाऱ्या प्रक्रियांसाठी किंवा कठोर वेळेचे बंधन नसलेल्या वर्कलोडसाठी योग्य आहे.
6. FCFS सर्व प्रोसेसना समान वागणूक देऊन आणि त्यांना रन करण्याची समान संधी देऊन निष्पक्षता प्रदान करते.
7. FCFS हमी देते की जोपर्यंत सिस्टमकडे सर्व प्रोसेस हाताळण्यासाठी पुरेसे संसाधने असतील तोपर्यंत प्रत्येक प्रोसेसला एक्झिक्युट करण्याची संधी मिळते.

तोटे (Disadvantages)

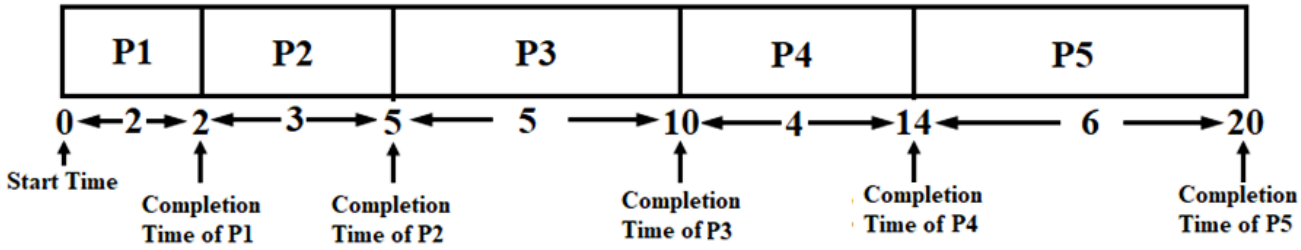
1. ते प्रोसेसच्या प्रायोरिटीचा किंवा बर्स्ट टाइमचा विचार करत नाही..
2. कमी एक्झिक्युशन वेळ असलेल्या प्रोसेसला जर वेटिंग टाइम जास्त असेल तर वाट बघावी लागते.
3. सीपीयू बाउंड प्रोसेससाठी अनुकूल आहे नंतर I/व बाउंड प्रोसेस.
4. कॉन्व्हॉय इफेक्ट (Convoy effect) मूळे ग्रस्त असते म्हणजेच कमी बर्स्ट टाइम असलेल्या प्रोसेसआधी जास्त बर्स्ट टाइम असलेली प्रोसेस आली तर सीपीयू आणि डिव्हाइस वापरावर परिणाम होतो.
5. FCFS अल्गोरिथम विशेषत: मल्टीप्रोग्रामिंग सिस्टमसाठी त्रासदायक आहे, जिथे प्रत्येक युजर्सला नियमित अंतराने सीपीयूचा वाटा मिळणे महत्वाचे आहे.

उदाहरण 1: खालील तक्त्यात दिलेल्या P1, P2, P3, P4, P5 प्रोसेसचा विचार करा, अराईवल टाइमसह आणि बर्स्ट टाइम दिलेला आहे. सरासरी टर्नअराउंड टाइम आणि सरासरी वेटिंग टाइम काढा. (Consider the processes P1, P2, P3, P4, P5 given in the below table, with arrival Time, and given Burst Time. Calculate average turnaround time and average waiting time.)

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	2
P2	1	3
P3	2	5
P4	3	4
P5	4	6

उत्तर:

गॅन्ट चार्ट (Gantt Chart): गॅन्ट चार्ट हा सीपीयूच्या शेड्यूलिंग प्रोसेसचे व्हिज्युअल रिप्रेझेन्टेशन आहे, जो सामान्यत: सीपीयूवर वेगवेगळ्या प्रोसेस किंवा टास्क कशी वाटली जातात हे दर्शविण्यासाठी वापरला जातो. हा एक हॉरिझंटल बार चार्ट आहे जिथे प्रत्येक प्रोसेस एका बारद्वारे दर्शविली जाते आणि बारची लांबी प्रोसेस रनिंग मध्ये असलेल्या वेळेशी संबंधित असते. हा चार्ट CPU युटिलायझेशन, थ्रूपुट , वेटिंग टाइम आणि टर्नअराउंड टाइम काढण्यास मदत करतो.



Turnaround Time(TAT) = Completion Time (CT) – Arrival Time(AT)

$P1_{TAT} = 2 - 0 = 2$

$P2_{TAT} = 5 – 1 = 4$

$P3_{TAT} = 10 – 2 = 8$

$P4_{TAT} = 14 – 3 = 11$

$P5_{TAT} = 20 – 4 = 16$

Average$_{TAT}$ = $\frac{(2+4+8+11+16)}{5} = \frac{41}{5} = 8.2$

Waiting Time (WT) = Turnaround Time(TAT) – Burst Time (BT)

$P1_{WT} = 2 - 2 = 0$

$P2_{WT} = 4 – 3 = 1$

$P3_{WT} = 8 – 5 = 3$

$P4_{WT} = 11 – 4 = 7$

$P5_{WTv} = 16 – 6 = 10$

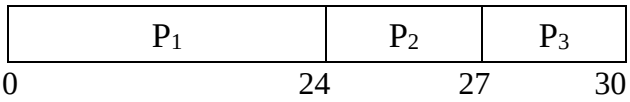
Average$_{WT}$ = $\frac{(0+1+3+7+10)}{5} = \frac{21}{5} = 4.2$

उदाहरण 2: खालील तक्त्यामध्ये दिलेल्या P1, P2, P3 या प्रोसेसेसचा विचार करा, ज्या त्याच क्रमाने एक्झिक्युशनसाठी येतात, अराईवल टाइम 0 सह, आणि बर्स्ट टाइम दिलेला आहे. सरासरी टर्नअराउंड टाइम आणि सरासरी वेटिंग टाइम काढा. (Consider the processes P1, P2, P3 given in the below table, arrives for execution in the same order, with Arrival Time 0, and given Burst Time. Calculate average turnaround time and average waiting time.)

उत्तर*:*

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	24
P2	0	3
P3	0	3

गॅन्ट चार्ट (Gantt Chart):



Turnaround Time(TAT) = Completion Time (CT) – Arrival Time(AT)

$P1_{TAT} = 24 - 0 = 24$

$P2_{TAT} = 27 – 0 = 27$

$P3_{TAT} = 30 – 0 = 30$

Average$_{TAT}$ = $\frac{(24+27+30)}{3} = \frac{81}{3} = 27$

Waiting Time (WT) = Turnaround Time(TAT) – Burst Time (BT)

$P1_{WT} = 24 - 24 = 0$

$P2_{WT} = 27 – 3 = 24$

$P3_{WT} = 30 – 3 = 27$

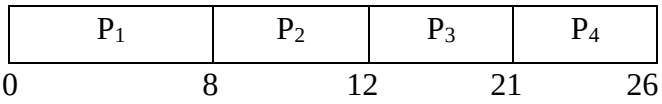
Average$_{WT}$ = $\frac{(0+24+27)}{3} = \frac{51}{3} = 17$

उदाहरण 3: खालील तक्त्यामध्ये दिलेल्या P1, P2, P3, P4 या प्रोसेसचा विचार करा, दिलेल्या अराईवल टाइम आणि बर्स्ट टाइमसह त्याच क्रमाने एक्झिक्युशनसाठी येतात. सरासरी टर्नअराउंड टाइम , सरासरी वेटिंग टाइम आणि थ्रूपुट काढा. (Consider the processes P1, P2, P3, P4 given in the below table, arrives for execution in the same order, with given Arrival Time and Burst Time. Calculate average turnaround time, average waiting time and throughput.)

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	8
P2	1	4
P3	2	9
P4	3	5

उत्तर*:*

गॅन्ट चार्ट (Gantt Chart):



PROCESS	TURN AROUND TIME	WAIT TIME
P1	8 – 0 = 8	0
P2	12 – 1 = 11	8 – 1 = 7
P3	21 – 2 = 19	12 – 2 = 10
P4	26 – 3 = 23	21 – 3 = 18

एकूण वेटिंग टाइम (Total Waiting Time) = 0 + 7 + 10 + 18 = 35

सरासरी वेटिंग टाइम (Average Waiting Time) = (Total Wait Time) / (Total number of processes)

= 35/4 = **8.75**

एकूण टर्नअराउंड टाइम (Total Turn Around Time) = 8 + 11 + 19 + 23 = 61 ms

सरासरी टर्नअराउंड टाइम (Average Turn Around time) = (Total Turn Around Time) / (Total number of processes) = 61/4 = **15.25 ms**

थ्रूपुट (Throughput) = Nos.of Jobs/Total of Jobs Burst Time = 4 jobs/26 sec = 0.15385 jobs/sec

3.3.2 शॉर्टेस्ट जॉब फर्स्ट (Shortest Job First - SJF)

सर्वात कमी बर्स्ट टाइम असलेल्या प्रोसेस प्रथम शेड्यूल केल्या जातात. जर दोन प्रोसेसेसचा बस्ट टाइम समान असेल, तर टाय तोडण्यासाठी FCFS वापरला जातो. हा एक नॉन-प्री-एम्प्टिव्ह आणि प्री-एम्प्टिव्ह शेड्यूलिंग अल्गोरिथम आहे. वेटिंग टाइम कमी करण्याचा सर्वोत्तम शेड्यूलिंग अल्गोरिथम आहे. बॅच सिस्टममध्ये इम्प्लिमेंट करणे सोपे आहे जिथे आवश्यक CPU वेळ आधीच माहित असते. इंटरॅक्टिव्ह सिस्टममध्ये इम्प्लिमेंट करणे अशक्य आहे जिथे आवश्यक CPU वेळ माहित नसते. प्रोसेसरला प्रोसेसला किती वेळ लागेल हे आधीच माहित असले पाहिजे. शॉर्टेस्ट जॉब फर्स्टच्या प्री-एम्प्टिव्ह मोडला शॉर्टेस्ट रिमेनिंग टाइम फर्स्ट (SRTF) म्हणतात.

फायदे (Advantages)

1. SJF ऑप्टिमल आहे आणि किमान सरासरी वेटिंग टाइमची हमी देतो.
2. हे इतर अल्गोरिथमसाठी एक मानक प्रदान करते कारण इतर कोणताही अल्गोरिथम त्यापेक्षा चांगले कार्य करत नाही.
3. हे बॅचमध्ये चालणाऱ्या जॉब्स साठी योग्य आहे, जिथे रन टाइम आधीच ज्ञात असतात.
4. सरासरी टर्नअराउंड टाइम (TAT) च्या बाबतीत SJF कदाचित ऑप्टिमल आहे.

तोटे (Disadvantages)

1. प्रोसेसेसचा बर्स्ट टाइम आधीच कळू शकत नाही.
2. जास्त बर्स्ट वेळ असलेल्या प्रोसेसेससाठी यामुळे स्टार्व्हेशन (Starvation) होते.
3. प्रोसेसेससाठी प्रायोरिटी निश्चित करता येत नाहीत.
4. जास्त बर्स्ट वेळ असलेल्या प्रोसेसेसचा रिस्पॉन्स टाइम कमी असतो.

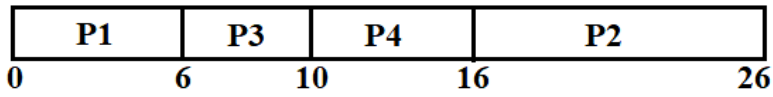
उदाहरण 1: समजा आपल्याकडे चार प्रोसेसेसचा संच आहे जो एकाच वेळी P1, P2, P3 आणि P4 या क्रमाने आला आहे. खालील टेबल मध्ये प्रत्येक प्रोसेसचा बर्स्ट टाइम मिलिसेकंदांमध्ये आहे. जर CPU शेड्युलिंग SJF नॉन-प्रीमेप्टिव्ह असेल, तर सरासरी वेटिंग टाइम आणि सरासरी टर्नअराउंड टाइम काढा. (Suppose that we have a set of four processes that have arrived at the same time in the order P1, P2, P3 and P4. The following table gives the burst time in milliseconds of each process. If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turnaround time.)

Process	Arrival time	Burst Time
P1	0	6
P2	1	10
P3	2	4
P4	3	6

उत्तर:

प्रोसेस P1 प्रथम आली, नंतर P1 ची एक्झिक्युशन सुरू झाले. P1 च्या एक्झिक्युशनचा दरम्यान, प्रोसेस P2, P3 आणि P4 आली तेव्हा P3 चा P2 आणि P4 पेक्षा सर्वात कमी बर्स्ट टाइम आणि म्हणून तो प्रथम एक्झिक्युट होते. शेवटी P4 आणि P2 पैकी P4 प्रथम एक्झिक्यु होते आणि शेवटी P2 एक्झिक्युट होते. खालील गँट चार्टद्वारे दिलेला आहे.

गॅन्ट चार्ट (Gantt Chart):



PROCESS	TURNAROUND TIME	WAIT TIME
P1	6-0 = 6	6 – 6 = 0
P2	26 – 1 = 25	25 – 10 = 15
P3	10 – 2 = 8	8 – 4 = 4
P4	16 – 3 = 13	13 – 6 = 7

सरासरी टर्नअराउंड टाइम $\textbf{Average}_{TAT} = \frac{(6+25+8+13)}{4} = \frac{52}{4} = 13$

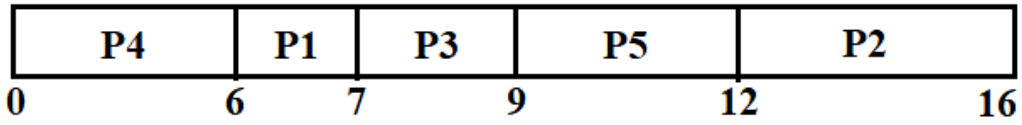
सरासरी वेटिंग टाइम $\textbf{Average}_{WT} = \frac{(0+15+4+7)}{4} = \frac{26}{4} = 6.5$

उदाहरण 2: खाली दिलेल्या 5 प्रोसेसेसचा संच विचारात घ्या ज्यांचे अराईवल टाइम आणि बर्स्ट टाइम खाली टेबल मध्ये दिले आहे. जर CPU शेड्युलिंग SJF नॉन-प्रीमेप्टिव्ह असेल, तर सरासरी वेटिंग टाइम आणि सरासरी टर्नअराउंड टाइम काढा. (Consider the set of 5 processes whose arrival time and burst time are given below and If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turnaround time.)

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

उत्तर:

गॅन्ट चार्ट (Gantt Chart):



आता, आपल्याला माहित आहे-

- टर्नअराउंड टाइम = कंप्लिशन टाइम - अराईवल टाइम
- वेटिंग टाइम = टर्नअराउंड टाइम - बर्स्ट टाइम

Process ID	Exit or completion time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

सरासरी टर्नअराउंड टाइम Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = **8 unit**

सरासरी वेटिंग टाइम Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = **4.8 unit**

3.3.3 शॉर्टेस्ट रिमेनिंग टाइम नेक्स्ट किंवा शॉर्टेस्ट रिमेनिंग टाइम फर्स्ट (Shortest Remaining Time Next - SRTN or Shortest Remaining Time First - SRTF)

शॉर्टेस्ट जॉब फर्स्ट (SJF) शेड्युलिंगची प्री-एम्प्टिव्ह आवृत्ती आहे, ज्याला शॉर्टेस्ट रिमेनिंग टाइम नेक्स्ट (SRTN) म्हणतात. SRTN मध्ये, पूर्ण होण्यासाठी कमीत कमी वेळ शिल्लक असलेली प्रोसेस रन करण्यासाठी निवडली जाते. रुनिंग प्रोसेस पूर्ण होईपर्यंत किंवा कमी वेळ शिल्लक असलेली नवीन प्रोसेस येईपर्यंत रन होत राहील. अशाप्रकारे, सर्वात कमी वेळात पूर्ण करू शकणाऱ्या प्रोसेसला नेहमीच प्राधान्य दिले जाते.

फायदे (Advantages)

1. सरासरी वेटिंग टाइम कमी करते: SRTF कमीत कमी उर्वरित एक्झिक्युशन टाइमसह प्रोसेसेसना प्राधान्य देऊन सरासरी वेटिंग टाइम कमी करते.
2. छोट्या प्रोसेसेससाठी कार्यक्षम: छोट्या प्रोसेस जलद पूर्ण होतात, ज्यामुळे एकूण सिस्टम प्रतिसाद सुधारतो.
3. वेळेच्या दृष्टीने क्रिटिकल सिस्टिमसाठी आदर्श: हे सुनिश्चित करते की टाइम सेन्सिटिव्ह प्रोसेस जलद एक्झिक्युट केल्या जातात.

तोटे (Disadvantages)

1. **लॉन्ग प्रोसेसेसची स्टार्व्हेशन (Starvation):** जर छोट्या प्रोसेस येत राहिल्या तर लॉन्ग प्रोसेस निश्चित काळासाठी विलंबित होऊ शकतात.
2. **बर्स्ट टाइमचा अंदाज लावणे कठीण:** प्रोसेसच्या बर्स्ट टाइमचा अचूक अंदाज लावणे आव्हानात्मक आहे आणि शेड्यूलिंग निर्णयांवर परिणाम करते.
3. **जास्त ओव्हरहेड:** वारंवार कॉन्टेक्स्ट स्विचिंगमुळे ओव्हरहेड वाढू शकते आणि सिस्टमची कार्यक्षमता मंदावू शकते.
4. **रिअल-टाइम सिस्टमसाठी योग्य नाही:** वारंवार प्रीएम्प्शनमुळे रिअल-टाइम टास्कना विलंब होऊ शकतो.

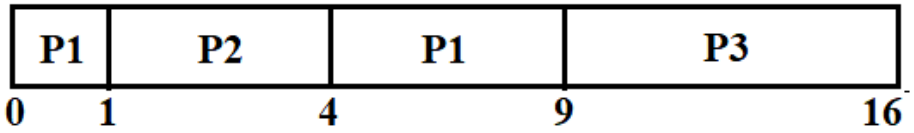
उदाहरण 1: P1, P2 आणि P3 या तीन प्रोसेसेससाठी अराईवल टाइम आणि बर्स्ट टाइमचा खालील टेबल विचारात घ्या. सरासरी टर्नअराउंड टाइम आणि सरासरी वेटिंग टाइम काढा. (Consider the following table of arrival time and burst time for three processes P1, P2 and P3. Calculate the average waiting time and average turnaround time.)

Process	Burst Time	Arrival Time
P1	6 ms	0 ms
P2	3 ms	1 ms
P3	7 ms	2 ms

उत्तर:

1. वेळ 0-1 (P1): P2 आल्यावर P1 1 मिलिसेकंद रन होते आणि त्याचा बर्स्ट टाइम 3 मिलिसेकंद आहे जो P1 चा एकूण शिल्लक वेळेपेक्षा कमी आहे म्हणजेच 5 मिलिसेकंद.
2. वेळ 1-4 (P2): P2 3 मिलिसेकंद रन होते (एकूण वेळ शिल्लक: 0 मिलिसेकंद) कारण त्याचा P1 आणि P3 मध्ये सर्वात कमी शिल्लक वेळ शिल्लक आहे.
3. वेळ 4-9 (P1): P1 5 मिलिसेकंद रन होते (एकूण वेळ शिल्लक: 0 मिलिसेकंद) कारण त्याचा P1 आणि P3 मध्ये सर्वात कमी शिल्लक वेळ शिल्लक आहे.
4. वेळ 9-16 (P3): P3 7 मिलिसेकंद रन होते (एकूण वेळ शिल्लक: 0 मिलिसेकंद) कारण त्याचा उर्वरित वेळ कमी आहे.

गॅन्ट चार्ट (Gantt Chart):



Process	कंप्लिशन टाइम Completion Time (CT)	टर्नअराउंड टाइम Turn Around Time (TAT)	वेटिंग टाइम Waiting Time (WT)
P1	9	9-0 = 9	9-6 = 3
P2	4	4-1 = 3	3-3 = 0
P3	16	16-2 = 14	14-7 = 7

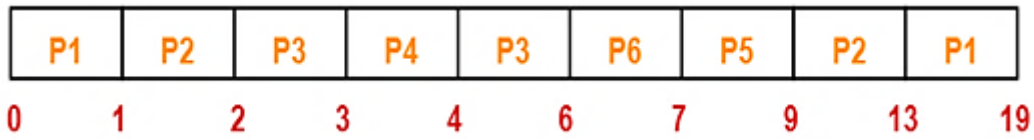
सरासरी टर्नअराउंड टाइम - Average Turnaround time = (9 + 14 + 3)/3 = **8.6 ms**

सरासरी वेटिंग टाइम - Average waiting time = (3 + 0 + 7)/3 = 10/3 = **3.33 ms**

उदाहरण 2: 6 प्रोसेसेसचा संच विचारात घ्या ज्यांची अराईवल टाइम आणि बर्स्ट टाइम खाली दिली आहे, सरासरी टर्नअराउंड टाइम आणि सरासरी वेटिंग टाइम काढा. (Consider the set of 6 processes whose arrival time and burst time are given below and calculate the average waiting time and average turnaround time.)

उत्तर:

गॅन्ट चार्ट (Gantt Chart):



Process Id	Completion time	Turn Around time	Waiting time
P1	19	19 – 0 = 19	19 – 7 = 12
P2	13	13 – 1 = 12	12 – 5 = 7
P3	6	6 – 2 = 4	4 – 3 = 1
P4	4	4 – 3 = 1	1 – 1 = 0
P5	9	9 – 4 = 5	5 – 2 = 3
P6	7	7 – 5 = 2	2 – 1 = 1

सरासरी टर्नअराउंड टाइम -Average Turn Around time = (19 + 12 + 4 + 1 + 5 + 2) / 6 = 43 / 6 = **7.17**

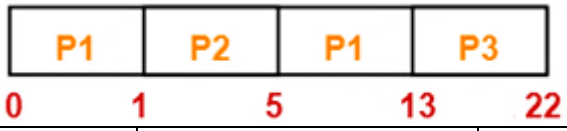
सरासरी वेटिंग टाइम - Average waiting time = (12 + 7 + 1 + 0 + 3 + 1) / 6 = 24 / 6 = **4**

उदाहरण 3: 3 प्रोसेसेसचा संच विचारात घ्या ज्यांची अराईवल टाइम आणि बर्स्ट टाइम खाली दिली आहे, सरासरी टर्नअराउंड टाइम आणि सरासरी वेटिंग टाइम काढा. (Consider the set of 3 processes whose arrival time and burst time are given below and calculate the average waiting time and average turnaround time.)

Process Id	Arrival time	Burst time
P1	0	9
P2	1	4
P3	2	9

उत्तर:

गॅन्ट चार्ट (Gantt Chart):



Process Id	Completion time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 9 = 4
P2	5	5 – 1 = 4	4 – 4 = 0
P3	22	22- 2 = 20	20 – 9 = 11

सरासरी टर्नअराउंड टाइम - Average Turn Around time = (13 + 4 + 20) / 3 = 37 / 3 = **12.33**

सरासरी वेटिंग टाइम - Average waiting time = (4 + 0 + 11) / 3 = 15 / 3 = **5**

3.3.4 राउंड रॉबिन शेड्यूलिंग (Round Robin (RR) Scheduling)

राउंड रॉबिन (RR) शेड्यूलिंग हे एक CPU शेड्यूलिंग अल्गोरिथम आहे जिथे प्रत्येक प्रोसेसला एक निश्चित टाइम स्लाईस किंवा "क्वांटम" वाटप केले जाते आणि जर प्रोसेस त्या वेळेत पूर्ण झाले नाही तर ते रेडी क्यूच्या शेवटी हलवले जाते कारण राउंड रॉबिन शेड्यूलिंग Fig. 3.3 मध्ये दाखवल्याप्रमाणे प्रीएम्प्टिव स्वरूपाचे आहे. आता, प्रोसेसर पुढील पुढील आलेल्या प्रोसेसला दिला जातो. हे फर्स्ट कम फर्स्ट सर्व्ह (FCFS) शेड्यूलिंग अल्गोरिथमचे प्रीएम्प्टिव आवृत्ती आहे.

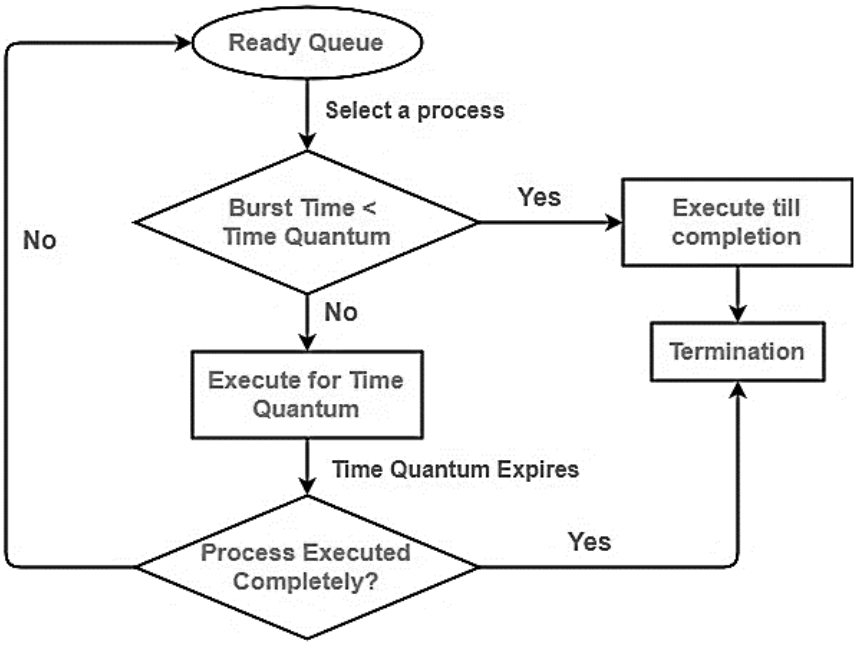


Fig.3.3: राउंड रॉबिन शेड्युलिंग अल्गोरिथमचा फ्लो डायग्राम (Flow diagram of Round Robin Scheduling Algorithm)

राऊंड-रोबिन शेड्यूलिंग कसे कार्य करते:

1. प्रोसेस रेडी क्यूत (एक वर्तुळाकार रांगेत) जोडल्या जातात.
2. सीपीयू क्यूतील पहिल्या प्रोसेसला अलोकेट केले जाते.
3. प्रोसेस निश्चित वेळेच्या क्वांटममध्ये रन होते.
4. जर प्रोसेस त्याच्या क्वांटममध्ये पूर्ण झाली तर ती क्यूतून काढून टाकली जाते.
5. जर प्रोसेस पूर्ण झाली नाही, तर ती क्यूच्या शेवटी ठेवली जाते आणि क्यूतील पुढील प्रोसेस सीपीयू दिली जाते.
6. क्यूतील सर्व प्रोसेस पूर्ण होईपर्यंत ही प्रक्रिया सुरू राहते.

राउंड रॉबिन शेड्युलिंगची प्रमुख वैशिष्ट्ये:

1. टाइम स्लाइसिंग (Time Slicing): प्रत्येक प्रोसेसला थोड्या प्रमाणात CPU वेळ दिला जातो, ज्याला क्वांटम किंवा टाइम स्लाइस म्हणतात.
2. सायक्लिक स्वरूप (Cyclic Nature): प्रोसेस वर्तुळाकार पद्धतीने एक्झिक्युट केल्या जातात, कारण त्यांचे क्वांटम संपल्यानंतर त्या रांगेच्या शेवटी ठेवल्या जातात.
3. प्रीएम्प्टिव्ह (Preemptive): वर्तमान प्रोसेस पूर्ण होण्यापूर्वी सिस्टम प्रोसेसेस मध्ये स्विच करू शकते, ज्यामुळे CPU वेळेचे योग्य वाटप सुनिश्चित होते.
4. निष्पक्षता (Fairness): सर्व प्रोसेसेसना एक्झिक्युट करण्याची समान संधी मिळते, ज्यामुळे कार्यांमध्ये निष्पक्षता वाढते.
5. व्यापकपणे वापरले जाणारे(Widely Used): RR शेड्यूलिंग हे ऑपरेटिंग सिस्टममध्ये, विशेषतः टाइम शेअरिंग सिस्टिम मध्ये एक लोकप्रिय अल्गोरिथम आहे.

टाइम क्वांटमच्या कमी केल्यास ,

1. कॉन्टेक्स्ट स्विचची संख्या वाढते
2. रिस्पॉन्स टाइम कमी होतो
3. स्टार्व्हेशनची शक्यता कमी होते

टाइम क्वांटमच्या वाढवल्यास

1. कॉन्टेक्स्ट स्विचची संख्या कमी होते
2. रिस्पॉन्स टाइम वाढतो
3. स्टार्व्हेशनची शक्यता वाढते

टाइम क्वांटमच्या वाढत्या मूल्यासह, राउंड रॉबिन शेड्युलिंग हे FCFS शेड्युलिंगमध्ये रूपांतरित होते. जेव्हा टाइम क्वांटम इन्फिनिटीकडे जातो, तेव्हा राउंड रॉबिन शेड्यूलिंग FCFS शेड्यूलिंग बनते. राऊंड रॉबिन शेड्यूलिंगची कामगिरी मोठ्या प्रमाणात टाइम क्वांटमच्या मूल्यावर अवलंबून असते. टाइम क्वांटमचे मूल्य असे असले पाहिजे की ते फार मोठे किंवा फारच लहान असायला नको.

फायदे (Advantages)

1. निष्पक्षता (Fairness): सर्व प्रोसेसेसना CPU चा योग्य वाटा मिळतो याची खात्री करते
2. साधेपणा (Simplicity): इम्प्लिमेंट करणे आणि समजणे सोपे आहे.
3. कमी वेटिंग टाइम (Low Waiting Time): प्रोसेससाठी सरासरी वेटिंग टाइम कमी करते.
4. इंटरॅक्टिव्ह सिस्टिमसाठी चांगले (Good for Interactive Systems): टाइम-शेअरिंग सिस्टिमसाठी योग्यरित्या अनुकूल आहे जेथे यूजर्स अनेक प्रोसेससह इंटरऍक्ट शकतात.

तोटे (Disadvantages)

1. वारंवार कॉन्टेक्स्ट स्विचिंगने ओव्हरहेड वाढू शकतो, विशेषतः जर क्वांटम खूप लहान असेल.
2. दीर्घकाळ रन होण्याऱ्या प्रोसेसेसना पूर्ण होण्यापूर्वी अनेक सायकलची आवश्यकता असू शकते, ज्यामुळे लहान प्रोसेसेसना विलंब होऊ शकतो.
3. यामुळे मोठ्या बर्स्ट टाइमच्या प्रोसेसेससाठी स्टार्व्हेशन होते कारण त्यांना सायकल अनेक वेळा रिपीट करावी लागते.

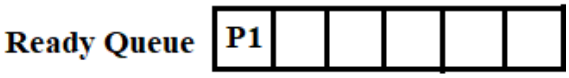
4. त्याची कामगिरी टाइम क्वांटमवर अवलंबून असते.
5. प्रोसेसेससाठी प्रायोरिटी निश्चित करता येत नाहीत.

उदाहरण 1: खालील 6 प्रक्रियांचा विचार करा: P1, P2, P3, P4, P5, आणि P6 त्यांच्या अराईवल टाइम आणि बर्स्ट टाइम दिला आहे. राउंड-रॉबिन शेड्युलिंग अल्गोरिथम (RR) साठी 4 युनिट्सच्या टाइम क्वांटमसह सरासरी टर्नअराउंड टाइम वेटिंग टाइम काढा. (Consider the following 6 processes: P1, P2, P3, P4, P5, and P6 with their arrival time and burst time as given below, Calculate the average waiting and turnaround times for the round-robin scheduling algorithm (RR) with a time quantum of 4 units.)

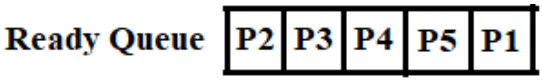
Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	6
P3	2	3
P4	3	1
P5	4	5
P6	6	4

उत्तर:

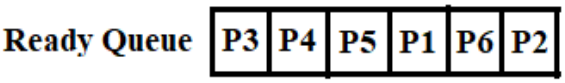
सुरुवातीला, रेडी क्यूत, प्रोसेस P1 4 युनिट्सच्या वेळेसाठी एक्झिक्युट केली जाईल. सुरुवातीला कोणत्याही प्रोसेस नसल्यामुळे, 5 युनिट्सच्या बर्स्ट टाइमसह प्रोसेस P1 ही रेडी क्यूत एकमेव प्रोसेस असेल.



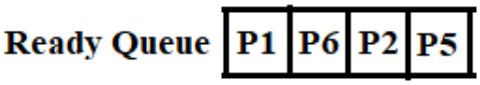
P1 च्या एक्झिक्युशनदरम्यान, P2, P3, P4 आणि P5 या चार आणखी प्रोसेस रेडी क्यूत येतात. P1 चे 4 युनिट झाल्यामुळे, उरलेल्या 1 युनिटसाठी, या आलेल्या प्रोसेस नंतर P1 रेडी क्यूत टाकला जाईल.



P2 च्या एक्झिक्युशनदरम्यान, P6 रेडी क्यूत आली. P2 पूर्ण झाले नसल्याने, P2 रेडी क्यूत टाकला जाईल.

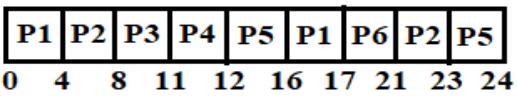


त्याचप्रमाणे, P3 आणि P4 पूर्ण झाले आहेत, परंतु P5 चा बर्स्ट टाइम 1 युनिट शिल्लक आहे. म्हणून, ते पुन्हा रेडी क्यूत टाकले जाईल.



पुढील प्रोसेस, P1, P6 आणि P2,एक्झिक्युट केल्या जातील. फक्त P5 मध्ये 1 युनिट बर्स्ट टाइम शिल्लक राहील आणि शेवटी तो एक्झिक्युट केली जाईल.

गॅन्ट चार्ट (Gantt Chart):



Processes	अराईवल टाइम Arrival Time(AT)	बर्स्ट टाइम Burst Time(BT)	टर्नअराउंड टाइम Turn Around Time(TAT)	वेटिंग टाइम Waiting Time(WT)
P1	0	5	17	12
P2	1	6	22	16
P3	2	3	9	6
P4	3	1	9	8
P5	4	5	20	15

सरासरी टर्नअराउंड टाइम - Average Turn Around time = (17 + 22 + 9 + 9 + 20) / 5 = 77 / 5 = **15.4**

सरासरी वेटिंग टाइम - Average waiting time = (12 + 16 + 6 +8 +15) / 5 = 57 / 5 = **11.4**

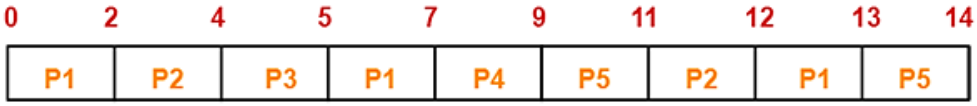
उदाहरण 2: खालील 5 प्रक्रियांचा विचार करा: P1, P2, P3, P4, P5, आणि P6 त्यांच्या अराईवल टाइम आणि बर्स्ट टाइम दिला आहे. राउंड-रॉबिन शेड्युलिंग अल्गोरिथम (RR) साठी 2 युनिट्सच्या टाइम क्वांटमसह सरासरी टर्नअराउंड टाइम वेटिंग टाइम काढा. (Consider the following 6 processes: P1, P2, P3, P4 and P5 with their arrival time and burst time as given below, Calculate the average waiting and turnaround times for the round-robin scheduling algorithm (RR) with a time quantum of 2 units.)

Process Id	Arrival time	Burst time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

उत्तर:

रेडी क्यू (Ready Queue) : P5, P1, P2, P5, P4, P1, P3, P2, P1

गॅन्ट चार्ट (Gantt Chart):



Process Id	Completion time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

सरासरी टर्नअराउंड टाइम - Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = **8.6**

सरासरी वेटिंग टाइम - Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = **5.8**

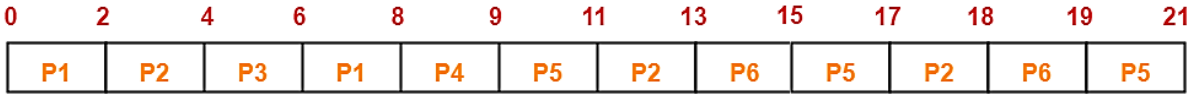
उदाहरण 3: खालील 6 प्रक्रियांचा विचार करा: P1, P2, P3, P4, P5, आणि P6 त्यांच्या अराईवल टाइम आणि बर्स्ट टाइम दिला आहे. राउंड-रॉबिन शेड्युलिंग अल्गोरिथम (RR) साठी 2 युनिट्सच्या टाइम क्वांटमसह सरासरी टर्नअराउंड टाइम वेटिंग टाइम काढा. (Consider the following 6 processes: P1, P2, P3, P4, P5, and P6 with their arrival time and burst time as given below, Calculate the average waiting and turnaround times for the round-robin scheduling algorithm (RR) with a time quantum of 2 units.)

Process Id	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

उत्तर:

रेडी क्यू (Ready Queue): P5, P6, P2, P5, P6, P2, P5, P4, P1, P3, P2, P1

गॅन्ट चार्ट (Gantt chart)



आपल्याला माहित आहे-

- टर्नअराउंड टाइम = कंप्लिशन टाइम - अराईवल टाइम
- वेटिंग टाइम = टर्नअराउंड टाइम - बर्स्ट टाइम

Process Id	Completion time	Turn Around time	Waiting time
P1	8	8 – 0 = 8	8 – 4 = 4
P2	18	18 – 1 = 17	17 – 5 = 12
P3	6	6 – 2 = 4	4 – 2 = 2
P4	9	9 – 3 = 6	6 – 1 = 5
P5	21	21 – 4 = 17	17 – 6 = 11
P6	19	19 – 6 = 13	13 – 3 = 10

सरासरी टर्नअराउंड टाइम - Average Turn Around time = (8 + 17 + 4 + 6 + 17 + 13) / 6
= 65 / 6 = **10.84 unit**

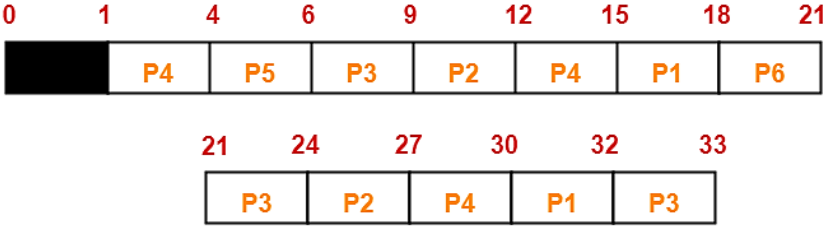
सरासरी वेटिंग टाइम - Average waiting time = (4 + 12 + 2 + 5 + 11 + 10) / 6 = 44 / 6 = **7.33 unit**

उदाहरण 4: खालील 6 प्रक्रियांचा विचार करा: P1, P2, P3, P4, P5, आणि P6 त्यांच्या अराईवल टाइम आणि बर्स्ट टाइम दिला आहे. राउंड-रॉबिन शेड्युलिंग अल्गोरिथम (RR) साठी 3 युनिट्सच्या टाइम क्वांटमसह सरासरी टर्नअराउंड टाइम वेटिंग टाइम काढा. (Consider the following 6 processes: P1, P2, P3, P4, P5, and P6 with their arrival time and burst time as given below, Calculate the average waiting and turnaround times for the round-robin scheduling algorithm (RR) with a time quantum of 3 units.)

Process Id	Arrival time	Burst time
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

उत्तर:

गॅन्ट चार्ट (Gantt chart)



Process Id	Completion time	Turn Around time	Waiting time
P1	32	32 – 5 = 27	27 – 5 = 22
P2	27	27 – 4 = 23	23 – 6 = 17
P3	33	33 – 3 = 30	30 – 7 = 23
P4	30	30 – 1 = 29	29 – 9 = 20
P5	6	6 – 2 = 4	4 – 2 = 2
P6	21	21 – 6 = 15	15 – 3 = 12

सरासरी टर्नअराउंड टाइम **-** Average Turn Around time = (27 + 23 + 30 + 29 + 4 + 15) / 6
= 128 / 6 = **21.33** unit

सरासरी वेटिंग टाइम - Average waiting time = (22 + 17 + 23 + 20 + 2 + 12) / 6 = 96 / 6 = **16** unit

3.3.5 प्रायोरिटी शेड्यूलिंग (Priority Scheduling)

प्रायोरिटी शेड्यूलिंग हे ऑपरेटिंग सिस्टमद्वारे त्यांच्या प्रायोरिटीवर आधारित प्रोसेस शेड्यूल करण्यासाठी वापरल्या जाणाऱ्या सर्वात सामान्य शेड्यूलिंग अल्गोरिथमपैकी एक आहे. प्रत्येक प्रोसेसला आवश्यक मेमरी , आवश्यक टाइम, इतर संसाधनांच्या गरजा किंवा सरासरी I/O आणि सरासरी CPU बर्स्ट टाईमचे गुणोत्तर यासारख्या निकषांवर आधारित प्रायोरिटी व्हाल्यू नियुक्त केले जाते. सर्वोच्च प्रायोरिटी असलेली प्रोसेस प्रथम एक्झिक्युशनसाठी निवडली जाते. जर समान प्रायोरिटी शेअरिंग करणाऱ्या अनेक प्रोसेस असतील, तर त्या प्रथम येणाऱ्यास प्राधान्य देण्याच्या दृष्टिकोनाचा अवलंब करून त्या ज्या क्रमाने आल्या त्या क्रमाने (First-Come, First-Served approach) शेड्यूल केल्या जातात. निवडलेली प्रोसेस पूर्ण होईपर्यंत किंवा ती प्रीएम्प्ट होईपर्यंत एक्झिक्युट केली जाते, हे शेड्यूलिंग प्रीएम्प्टिव्ह आहे की नॉन-प्रीएम्प्टिव्ह यावर अवलंबून असते. सर्वोच्च प्रायोरिटी असलेल्या प्रोसेससाठी वेटिंग टाइम प्रीएम्प्टिव्ह मोडमध्ये नेहमीच शून्य असते. सर्वोच्च प्रायोरिटी असलेल्या प्रोसेससाठी वेटिंग टाइम नॉन-प्रीएम्प्टिव्ह मोडमध्ये शून्य असू शकत नाही. प्रीएम्प्टिव्ह आणि नॉन-प्रीएम्प्टिव्ह मोडमध्ये प्रायोरिटी शेड्यूलिंग खालील परिस्थितीत अगदी सारखेच असते –

1. सर्व प्रोसेसचा अराईवल टाइम सारखाच असला तर .
2. सर्व प्रोसेस उपलब्ध असल्या तर .

नॉन-प्रीएम्प्टिव्ह प्रायोरिटी शेड्युलिंग (Non-Preemptive Priority Scheduling)

नॉन-प्रीएम्प्टिव्ह प्रायोरिटी शेड्युलिंगमध्ये, सीपीयू रनिंग प्रोसेस काढून टाकली जात नाही. जरी उच्च-प्रायोरिटी प्रोसेस आली तरी, सध्या रनिंग मध्ये असलेली प्रोसेस प्रथम पूर्ण होईल.

प्रीएम्प्टिव्ह प्रायोरिटी शेड्यूलिंग (Preemptive Priority Scheduling)

प्रीएम्प्टिव्ह प्रायोरिटी शेड्युलिंगमध्ये, जर उच्च प्रायोरिटी असलेली नवीन प्रोसेस आली तर CPU सध्या रनिंग करीत असलेली प्रोसेस काढून टाकली जाते.

फायदे (Advantages)

1. एक्झिक्युशन करणे सोपे आहे कारण शेड्यूलरला कोणतीही पूर्व कॅल्क्युलेशन करण्याची आवश्यकता नसते.
2. २. एकदा सीपीयूने प्रोसेसची संबंधित प्रायोरिटी परिभाषित केली की एक्झिक्युशनचा क्रम सहजपणे अंदाज लावता येतो.
3. उच्च प्रायोरिटी प्रोसेस जवळजवळ त्वरित पूर्ण केल्या जातात..
4. प्रायोरिटी शेड्यूलिंग विशेषतः अशा सिस्टिममध्ये उपयुक्त आहे ज्यांच्या स्वतःच्या गरजा असलेल्या विविध प्रोसेस असतात.

तोटे (Disadvantages)

1. स्टॅटिक प्रायोरिटी सिस्टीममध्ये, कमी प्रायोरिटी प्रोसेसना अनिश्चित काळासाठी वाट पहावी लागू शकते कारण सिस्टम उच्च प्रायोरिटी प्रोसेस एक्झिक्युट करण्यात व्यस्त असते. यामुळे स्टॅगनेशन (Stagnation) होतो.

2. डायनॅमिक प्रायोरिटी स्टॅगनेसची समस्या सोडवते. तथापि, सिस्टमनुसार डायनॅमिकली प्रायोरिटी व्हॅल्यूज अपडेट करण्याच्या जोडलेल्या लॉजिकसाठी अतिरिक्त CPU सायकलची आवश्यकता असते आणि त्यामुळे सिस्टमवरील भार वाढतो.
3. नॉन-प्रीएम्प्टिव्ह प्रायोरिटी शेड्यूलिंगमध्ये, अनेकदा मोठी प्रक्रिया लहान प्रोसेसना बराच काळ वाट पाहत ठेवते.
4. प्रीएम्प्टिव्ह प्रायोरिटी शेड्यूलिंगमध्ये, कमी प्रायोरिटी प्रोसेस वारंवार कॉन्टेक्स्ट स्विचची आवश्यकता असलेल्या उच्च प्रायोरिटी प्रोसेसच्या अधूनमधून स्ट्रीमद्वारे वारंवार प्रीएम्प्टिव्ह होऊ शकते.

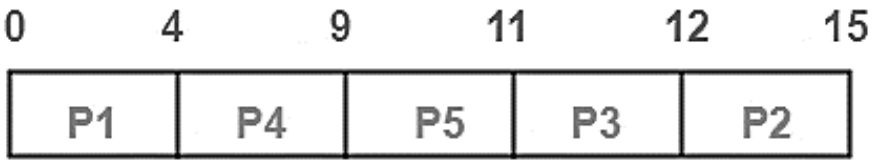
उदाहरण 1: खालील प्रक्रियांचा विचार करा: P1, P2, P3, P4, आणि P5, त्यांच्या अराईवल टाइम आणि बर्स्ट टाइम दिला आहे. जर CPU शेड्युलिंग अल्गोरिथम नॉन-प्रायोरिटी शेड्यूलिंग असेल, तर सरासरी वेटिंग टाइम आणि सरासरी टर्नअराउंड टाइम काढा. (समजा, मोठी संख्या ही उच्च प्रायोरिटी दर्शवते)

(Consider the set of 5 processes whose arrival time and burst time are given below, If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turnaround time. *Consider higher number represents higher priority*)

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

उत्तर:

गॅन्ट चार्ट (Gantt chart)



Process Id	Completion time	Turn Around time	Waiting time
P1	4	4 – 0 = 4	4 – 4 = 0
P2	15	15 – 1 = 14	14 – 3 = 11
P3	12	12 – 2 = 10	10 – 1 = 9
P4	9	9 – 3 = 6	6 – 5 = 1
P5	11	11 – 4 = 7	7 – 2 = 5

सरासरी टर्नअराउंड टाइम- Average Turn Around time = (4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = **8.2 unit**

सरासरी वेटिंग टाइम-Average waiting time = (0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = **5.2 unit**

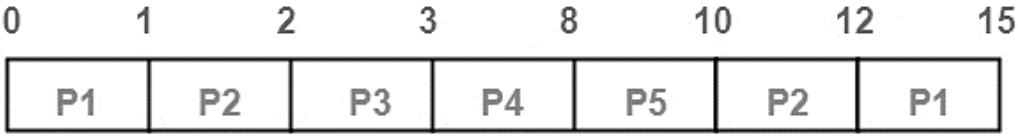
उदाहरण 1: खालील प्रक्रियांचा विचार करा: P1, P2, P3, P4, आणि P5, त्यांच्या अराईवल टाइम आणि बर्स्ट टाइम दिला आहे. जर CPU शेड्युलिंग अल्गोरिथम प्रायोरिटी शेड्यूलिंग असेल, तर सरासरी वेटिंग टाइम आणि सरासरी टर्नअराउंड टाइम काढा. (समजा, मोठी संख्या ही उच्च प्रायोरिटी दर्शवते)

(Consider the set of 5 processes whose arrival time and burst time are given below, If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turnaround time. *Consider higher number represents higher priority*)

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

उत्तर:

गॅन्ट चार्ट (Gantt chart)



आपल्याला माहित आहे-

- टर्नअराउंड टाइम = कंप्लिशन टाइम - अराईवल टाइम
- वेटिंग टाइम = टर्नअराउंड टाइम - बर्स्ट टाइम

Process Id	Completion time	Turn Around time	Waiting time
P1	15	15 – 0 = 15	15 – 4 = 11
P2	12	12 – 1 = 11	11 – 3 = 8
P3	3	3 – 2 = 1	1 – 1 = 0
P4	8	8 – 3 = 5	5 – 5 = 0
P5	10	10 – 4 = 6	6 – 2 = 4

सरासरी टर्नअराउंड टाइम-Average Turn Around time = (15 + 11 + 1 + 5 + 6) / 5 = 38 / 5 = **7.6 unit**

सरासरी वेटिंग टाइम- Average waiting time = (11 + 8 + 0 + 0 + 4) / 5 = 23 / 5 = **4.6 unit**

3.3.6 मल्टीलेव्हल क्यू शेड्यूलिंग (Multilevel Queue Scheduling)

मल्टीलेव्हल क्यू शेड्युलिंग अल्गोरिथम रेडी क्यूला अनेक वेगवेगळ्या क्यूमध्ये विभाजित करतो. प्रोसेस कायमस्वरूपी एका क्यूला दिल्या जातात, सामान्यतः प्रोसेसच्या काही गुणधर्मांवर आधारित, जसे की मेमरी आकार (Memory size), प्रोसेस प्रायोरिटी किंवा प्रोसेसचे प्रकार. प्रत्येक क्यूचे स्वतःचे शेड्युलिंग अल्गोरिथम असते.

पाच क्यूसह मल्टीलेव्हल क्यू-शेड्युलिंग अल्गोरिथमचे उदाहरण पाहूया जे Fig. 3.4 मध्ये दर्शविले आहे.:

1. सिस्टम प्रोसेस
2. इंटरॅक्टिव्ह प्रोसेस
3. इंटरॅक्टिव्ह एडिटिंग प्रोसेस
4. बॅच प्रोसेस
5. स्टुडन्ट प्रोसेस

प्रत्येक क्यूला कमी-प्रायोरिटी असलेल्या क्यूपेक्षा पूर्ण प्रायोरिटी असते. उदाहरणार्थ, बॅच क्यूतील कोणतीही प्रोसेस सिस्टम प्रोसेस, इंटरॅक्टिव्ह प्रोसेस आणि इंटरॅक्टिव्ह एडिटिंग प्रोसेस साठीच्या क्यू रिकाम्या असल्याशिवाय रन होऊ शकत नाही. जर बॅच प्रोसेस रन होत असताना इंटरॅक्टिव्ह एडिटिंग प्रोसेस रेडी क्यूत प्रवेश केली तर बॅच प्रोसेस प्रीएम्प्ट केली जाते.

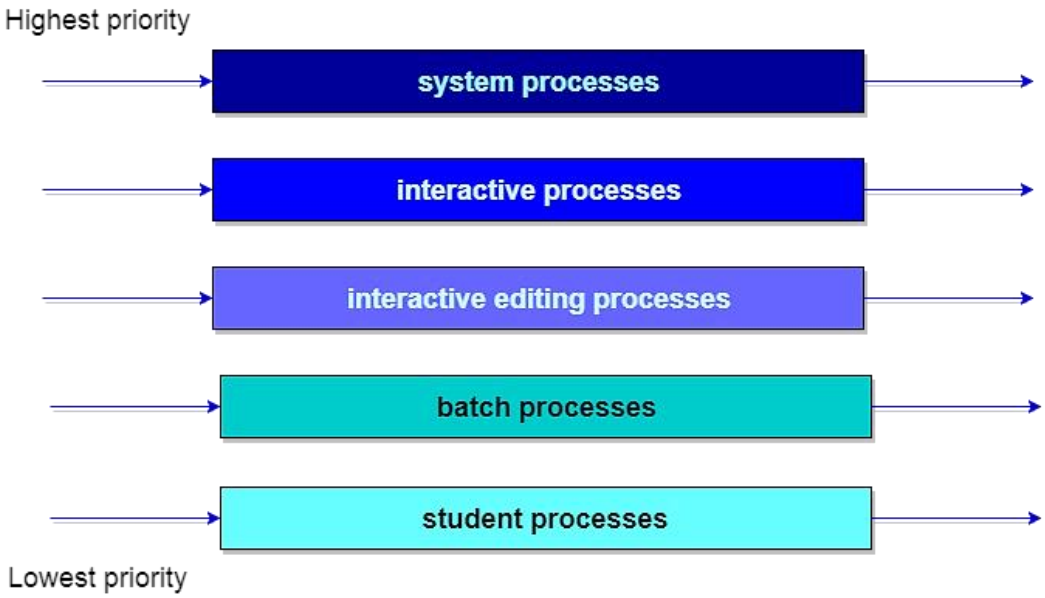


Fig. 3.4: मल्टीलेव्हल क्यू शेड्यूलिंग (Multilevel Queue Scheduling)

3.4 डेडलॉक (Deadlock)

डेडलॉक म्हणजे अशी परिस्थिती जिथे प्रोसेसेसचा एक संच ब्लॉक केला जातो कारण प्रत्येक प्रोसेसने एक संसाधन (Resource) अधिग्रहित केली असते आणि दुसऱ्या प्रोसेसद्वारे अधिग्रहित केलेल्या दुसऱ्या संसाधनाची वाट पाहत असते. उदाहरणार्थ, खालील आकृतीमध्ये, प्रोसेस 1 ने संसाधन 1 अधिग्रहित केला आहे आणि प्रोसेस 2 द्वारे अधिग्रहित केलेल्या संसाधन 2 ची वाट पाहत आहे आणि प्रोसेस 2 Fig.3.5 मध्ये दाखवल्याप्रमाणे संसाधन 1 ची वाट पाहत आहे.

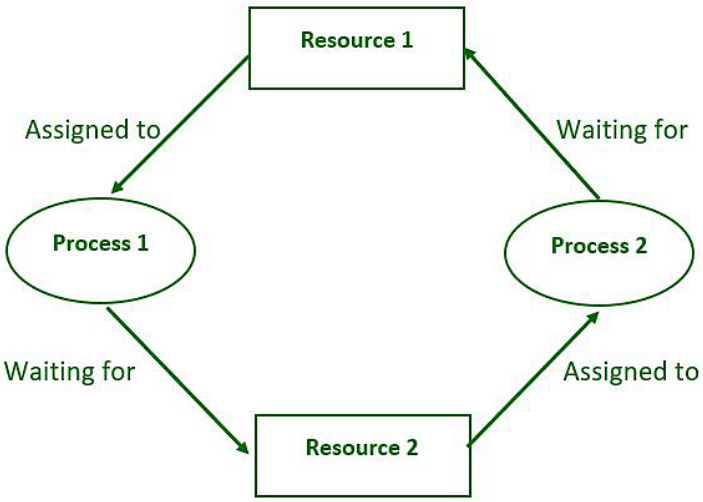


Fig.3.5: ऑपरेटिंग सिस्टममध्ये डेडलॉक (Deadlock in Operating System)

3.4.1 डेडलॉकचे सिस्टम मॉडेल (System model of Deadlock)

प्रोसेस सिंक्रोनाइझेशनच्या अभावामुळे दोन अत्यंत गंभीर परिस्थिती उद्भवू शकतात: डेडलॉक किंवा स्टार्व्हेशन. डेडलॉक ही मल्टीप्रोग्राम सिस्टमची समस्या आहे. डेडलॉक म्हणजे सिस्टम संसाधनांसाठी पूर्ण होणाऱ्या प्रोसेसेसच्या संचाचे कायमचे ब्लॉकिंग म्हणून परिभाषित केले जाऊ शकते. डेडलॉक हा स्टार्व्हेशनपेक्षा अधिक गंभीर असते. डेडलॉक फाइल्स, प्रिंटर, डेटाबेस, डिस्क, टेप ड्राइव्ह, मेमरी, सीपीयू सायकल इत्यादी शेअर करण्यायोग्य संसाधनांवर होऊ शकतो. एखादी प्रोसेस जर एखाद्या विशिष्ट घटनेची वाट पाहत असेल जी घडणार असते तर ती डेडलॉक स्थितीत असते. सिस्टम डेडलॉकमध्ये, एक किंवा अधिक प्रोसेस डेडलॉक असतात. संगणक सिस्टिम ही मर्यादित/अमर्यादित संख्येचा संसाधनांचा संग्रह असते. ही संसाधने अनेक कॉम्पेटिंग प्रोसेसमध्ये वाटप केली जातात.

संसाधने दोन प्रकारची असतात:

1. पुनर्वापरयोग्य संसाधने (Reusable resources)
2. कंझुमेबल संसाधने (Consumable resources)

पुनर्वापरयोग्य संसाधन एका वेळी फक्त एकाच प्रोसेसद्वारे वापरले जाते आणि प्रोसेस वापरल्यानंतर संसाधनाच ताबा सोडू शकते. प्रोसेसर, I/O चॅनेल, I/O डिव्हाइस, फाइल्स, डेटाबेस, प्रायमरी आणि सेकंडरी मेमरी, सेमाफोर ही पुनर्वापरयोग्य संसाधनाची उदाहरणे आहेत. कंझुमेबल संसाधन म्हणजे जे तयार आणि नष्ट केले जाऊ शकते. विशिष्ट प्रकारच्या कंझुमेबल संसाधनांच्या संख्येवर कोणतीही मर्यादा नसते. इंटरप्ट, मेसेज, सिग्नल आणि I/O बफरमधील मेसेजेस ही कंझुमेबल संसाधनांची उदाहरणे आहेत.

प्रोसेस पुढील क्रमाने संसाधनांचा वापर करते:

1. संसाधनासाठी विनंती (Request for resource)
2. संसाधनाचा वापर (Use of resource)
3. संसाधनाच ताबा सोडणे (Resource release)

संसाधनाची विनंती करण्यासाठी प्रोसेस सिस्टम कॉलचा वापर करते. प्रोसेसला संसाधन वाटप केल्यानंतर, ती संसाधन वापरते किंवा ऑपरेट करते. प्रोसेस संसाधन वापरल्यानंतर संसाधन सोडते. विनंती केल्यावर संसाधन उपलब्ध नसल्यास, विनंती करण्याऱ्या प्रोसेसला प्रतीक्षा करणे भाग पडते.

3.4.2 डेडलॉक होण्यासाठी आवश्यक अटी (Necessary conditions leading to deadlock)

खालील चार आवश्यक अटी एकाच वेळी आल्यास डेडलॉक निर्माण होऊ शकतो.

1. **म्युच्युअल एक्सक्लूजन** (**Mutual Exclusion)** : एक किंवा एकापेक्षा जास्त संसाधने नॉन-शेअर करण्यायोग्य असतात म्हणजे एका वेळी फक्त एक प्रोसेस वापरू शकते.
2. **होल्ड आणि वेट** (**Hold and Wait)**: एक प्रोसेस किमान एक संसाधन अधिग्रहित करून दुसऱ्या संसाधनाची वाट पाहत असते.
3. **नो प्री-एम्प्शन** (**No Pre-emption)**: प्रोसेसने संसाधन रिलीज केल्याशिवाय संसाधन घेतले जाऊ शकत नाही. म्हणजे प्रोसेस, जी एकदा शेड्यूल केली गेली की ती पूर्ण होईपर्यंत एक्झिक्युट केली जाते आणि शेड्यूलर, दरम्यान इतर कोणतीही प्रोसेस शेड्यूल करू शकत नाही.
4. **सर्क्युलर वेट** (**Circular Wait)**: प्रक्रियांचा संच सर्क्युलर स्वरूपात एकमेकांची वाट पाहत असते याचा अर्थ सर्व प्रक्रिया सायक्लिक पद्धतीने संसाधनांची वाट पाहत असतात जेणेकरून शेवटची प्रोसेस पहिल्या प्रोसेसद्वारे अधिग्रहित असलेल्या संसाधनाची वाट पाहत असते.

3.4.3 डेडलॉक हाताळणी (Deadlock Handling)

डेडलॉक हाताळण्यासाठी खालील विविध रणनीती आहेत-

1. डेडलॉक प्रतिबंध (Deadlock Prevention)
2. डेडलॉक टाळणे (Deadlock Avoidance)
3. डेडलॉक डिटेक्शन आणि रिकव्हरी (Deadlock Detection and Recovery)
4. डेडलॉक इग्नोरन्स (Deadlock Ignorance)

1. डेडलॉक प्रतिबंध (Deadlock Prevention)

चार आवश्यक अटींपैकी कमीतकमी एकाला प्रतिबंधित करून डेडलॉक्स प्रतिबंधित केले जाऊ शकतात:

1. **म्युच्युअल एक्सक्लूजन:** शेअर्ड संसाधने जसे की रीड ओन्ली फायली डेडलॉककडे नेत नाहीत. दुर्दैवाने, प्रिंटर आणि टेप ड्राइव्हसारख्या काही संसाधनांना एकाच प्रोसेसद्वारे विशेष ऍक्सेस आवश्यक असतो.
2. **होल्ड अँड वेट:** ही स्थिती टाळण्यासाठी प्रोसेसेसना एकाच वेळी एक किंवा अधिक संसाधने धरून ठेवण्यापासून रोखले पाहिजे आणि एकाच वेळी एक किंवा अधिक इतरांची वाट पाहत राहावे लागते.
3. **नो प्री-एम्प्शन** : प्रोसेसेसना संसाधन वाटपाचे प्रीएम्प्शन शक्य असल्यास डेडलॉकची ही स्थिती रोखू शकते.
4. **सर्क्युलर वेट** : सर्क्युलर वेट टाळण्याचा एक मार्ग म्हणजे सर्व संसाधनांची संख्या करणे आणि प्रोसेसेना केवळ काटेकोरपणे वाढत्या (किंवा कमी होत जाणाऱ्या) क्रमाने संसाधनांची विनंती करणे आवश्यक आहे.

2. डेडलॉक टाळणे (Deadlock Avoidance)

डेडलॉक टाळण्यामध्ये, ऑपरेटिंग सिस्टम प्रत्येक टप्प्यावर सिस्टम सुरक्षित स्थितीत किंवा असुरक्षित स्थितीत आहे की नाही हे तपासते. सिस्टम सुरक्षित स्थितीत येईपर्यंत ही प्रोसेस चालू राहते. एकदा सिस्टम असुरक्षित स्थितीत गेल्यावर, OS ला एक पाऊल मागे घ्यावे लागते. सोप्या शब्दात, OS प्रत्येक वाटपाचे पुनरावलोकन करते जेणेकरून वाटपामुळे सिस्टीममध्ये डेडलॉक निर्माण होऊ नये.

3. डेडलॉक डिटेक्शन आणि रिकव्हरी (Deadlock Detection and Recovery)

डेडलॉक टाळता येत नसतील तर, त्यांचा शोध घेणे आणि काही मार्गाने रिकव्हर करणे हा दुसरा मार्ग आहे. या रणनीतीमध्ये डेडलॉक होईपर्यंत प्रतीक्षा करणे समाविष्ट आहे. डेडलॉक झाल्यानंतर, सिस्टमची स्थिती रिकव्हर केली जाते. या दृष्टिकोनात मुख्य आव्हान हे डेडलॉक शोधणे आहे.

4. डेडलॉक इग्नोरन्स (Deadlock Ignorance)

या स्ट्रॅटेजीमध्ये डेडलॉकच्या संकल्पनेकडे दुर्लक्ष करणे आणि ते अस्तित्वात नाही असे गृहीत धरणे समाविष्ट आहे. ही रणनीती डेडलॉक हाताळताना अतिरिक्त ओव्हरहेड टाळण्यास मदत करते. विंडोज आणि लिनक्स ही स्ट्रॅटेजी वापरतात आणि ही सर्वात जास्त वापरली जाणारी पद्धत आहे.

3.4.4 बँकर्स अल्गोरिथम (Banker's Algorithm)

बँकरचे अल्गोरिथम हे डेडलॉक टाळणे किंवा शोधण्याचे अल्गोरिथम आहे आणि त्याला असे नाव देण्यात आले आहे कारण हे अल्गोरिथम बँकिंग सिस्टममध्ये कर्ज दिले जाऊ शकते की नाही हे निर्धारित करण्यासाठी वापरले जाते. जेव्हा जेव्हा एखादी नवीन प्रोसेस तयार केली जाते, तेव्हा तिला आवश्यक असलेल्या प्रत्येक संसाधन प्रकाराच्या जास्तीत जास्त किती इन्स्टेन्ससची संख्या अचूकपणे सांगणे आवश्यक असते. ही संख्या सिस्टममधील संसाधनांच्या एकूण संख्येपेक्षा जास्त नसावी. आता, जेव्हा नवीन प्रोसेस संसाधनांची विनंती करते, तेव्हा सिस्टमने गणना केली पाहिजे की विनंती केलेल्या संसाधनांचे वाटप संगणक सिस्टिमला सुरक्षित स्थितीत ठेवेल की नाही. तसे असल्यास, प्रोसेसला विनंती केलेल्या संसाधनांचे वाटप केले जाईल, अन्यथा काही इतर प्रोसेस विनंती केलेली संसाधने रिलीझ होईपर्यंत प्रतीक्षा करावी लागेल. या पद्धतीचे पालन करून, बँकरचा अल्गोरिथम डेडलॉक टाळतो आणि संसाधनांचे सुरक्षितपणे वाटप करतो. बँकरच्या अल्गोरिथमचे इम्प्लिमेंट करण्यासाठी वापरल्या जाणाऱ्या डेटा स्ट्रक्चर्स आहेत:

- **अवेलेबल (Available)**: हा m लांबीचा ॲरे आहे. तो प्रत्येक प्रकारच्या उपलब्ध संसाधनांची संख्या दर्शवितो. जर अवेलेबल[j] = k असेल, तर संसाधन प्रकार Rj चे k इन्स्टन्स उपलब्ध आहेत.
- **मॅक्स** (**Max)**: हा एक n x m मॅट्रिक्स आहे जो प्रोसेस विनंती करू शकणाऱ्या प्रत्येक संसाधनाच्या जास्तीत जास्त संख्येचे प्रतिनिधित्व करतो. जर मॅक्स[i][j] = k असेल, तर प्रोसेस Pi संसाधन प्रकार Rj च्या किमान k इन्स्टन्सची विनंती करू शकते.
- **अलोकेशन** (**Allocation)**: हा एक n x m मॅट्रिक्स आहे जो प्रत्येकप्रोसेसला सध्या अलोकेट केलेल्या प्रत्येक प्रकारच्या संसाधनांची संख्या दर्शवितो. जर अलोकेशन[i][j] = k असेल, तर प्रोसेस Pi सध्या संसाधन प्रकार Rj च्या k इन्स्टन्सचे वाटप करते.
- **नीड** (**Need)**: हा एक द्विमितीय ॲरे आहे. हा एक n x m मॅट्रिक्स आहे जो प्रत्येक प्रोसेसच्या उर्वरित संसाधन गरजा दर्शवितो. जर गरज[i][j] = k असेल, तर प्रोसेस Pi ला त्याचे टास्क पूर्ण करण्यासाठी संसाधन प्रकार Rj च्या k अधिक इन्स्टन्सची आवश्यकता असू शकते.

$$Need[i][j] = Max[i][j] - Allocation [i][j]$$

बँकरच्या अल्गोरिथममध्ये दोन अल्गोरिथम असतात:

1. सेफ्टी अल्गोरिथम (Safety algorithm)
2. संसाधन विनंती अल्गोरिथम (Resource request algorithm)

1. सेफ्टी अल्गोरिथम (Safety algorithm)

सेफ्टी अल्गोरिथम एक अल्गोरिथम आहे जो सिस्टम त्याच्या सुरक्षित स्थितीत आहे की नाही हे शोधण्यासाठी वापरला जातो. अल्गोरिथम खालीलप्रमाणे आहे:

स्टेप 1: Work आणि Finish अनुक्रमे लांबी m आणि n चे वेक्टर आहे.

सुरुवातीला, Work = Available

Finish[i] = false for i = 0, 1, ... , n - 1.

याचा अर्थ, सुरुवातीला कोणतीही प्रोसेस पूर्ण झालेली नाही आणि उपलब्ध संसाधनांची संख्या Available अॅरेद्वारे दर्शविली जाते.

स्टेप 2: असा इंडेक्स i शोधा की दोन्ही

Finish[i] ==false

Needi <= Work

जर असा कोणताही i नसेल, तर स्टेप 4 वर जा.

याचा अर्थ, आपल्याला अशी अपूर्ण प्रोसेस शोधण्याची आवश्यकता आहे जिच्या गरजा उपलब्ध संसाधने पूर्ण करू शकतील. जर अशी कोणतीही प्रोसेस अस्तित्वात नसेल, तर फक्त चरण स्टेप 4 वर जा.

स्टेप 3: खालील गोष्टी करा:

Work = Work + Allocation

Finish[i] = true

स्टेप 2 वर जा.

जेव्हा एखादी अपूर्ण प्रोसेस आढळते, तेव्हा संसाधनांचे वाटप केले जाते आणि प्रोसेस पूर्ण झाली असे चिन्हांकित केले जाते. नंतर, इतर सर्व प्रोसेसेससाठी तेच तपासण्यासाठी लूपची पुनरावृत्ती केली जाते.

स्टेप 4: जर **Finish[i] == True** सर्व **i** साठी असेल**,** तर सिस्टम सुरक्षित स्थितीत आहे.म्हणजेच जर सर्व प्रक्रिया पूर्ण झाल्या**,** तर सिस्टम सुरक्षित स्थितीत आहे.

2. संसाधन विनंती अल्गोरिथम (Resource request algorithm)

आता पुढील अल्गोरिथम म्हणजे रिसोर्स-रिक्वेस्ट अल्गोरिथम आणि तो प्रामुख्याने विनंत्या सुरक्षितपणे मंजूर करता येतात की नाही हे ठरवण्यासाठी वापरला जातो. Pi प्रोसेससाठी Request***i*** ला रिक्वेस्ट वेक्टर म्हणून समजा. जर Request***i***[j]==k असेल, तर प्रोसेस Pi ला रिसोर्स प्रकार Rj चा k इन्स्टन्स हवा आहे. प्रोसेस Pi संसाधनांसाठी विनंती करते, तेव्हा खालील कृती केल्या जातील:

स्टेप 1: जर Request***i*** <= Need***i*** असेल, तर स्टेप 2 वर जा; अन्यथा प्रोसेसने त्याच्या कमाल दाव्याची मर्यादा ओलांडली असल्याने एरर निर्माण करा.

स्टेप 2: जर Request***i*** <= Available***i*** असेल तर स्टेप 3 वर जा; अन्यथा संसाधने उपलब्ध नसल्यामुळे Pi ला वाट पहावी लागेल.

स्टेप 3: आता आपण असे गृहीत धरू की Pi प्रोसेस करण्यासाठी संसाधने नियुक्त केली आहेत आणि खालील स्टेप पार पाडू: Available= Available-Request***i***;

Allocation***i***=Allocation***i*** +Request***i***;

Need***i*** =Need***i*** - Request***i***;

जर परिणामी संसाधन वाटप स्थिती सुरक्षित असल्याचे दिसून आले, तर ट्रान्झॅक्शन पूर्ण होतो आणि प्रोसेस Pi ला त्याचे संसाधने वाटप केली जातात. या प्रकरणात, जर नवीन स्थिती असुरक्षित असेल, तर Pi Requesti ची वाट पाहतो आणि जुनी संसाधन वाटप स्थिती रिस्टोअर्ड केली जाते.

उदाहरण 1: एका सिस्टिमचा विचार करा ज्यामध्ये पाच प्रोसेस P1, P2, P3, P4, P5 आणि तीन संसाधन प्रकार A, B आणि C आहेत. खालील संसाधन प्रकार आहेत: A चे 10, B चे 5 आणि संसाधन प्रकार C चे 7 इन्स्टन्स उपलब्ध आहेत. सिस्टिम सेफ स्टेट मध्ये आहे कि नाही शोध. (Consider a system that contains five processes P1, P2, P3, P4, P5 and the three resource types A, B and C. Following are the resources types: A has 10, B has 5 and the resource type C has 7 instances available. Find either system is in safe state or not.)

Process	**Allocation** A	B	C	**Max** A	B	C	**Available (Total Available- Total Allocated) A=10 B=5 C=7**
P1	0	1	0	7	5	3	10-7=3 5-2=3 7-5=2
P2	2	0	0	3	2	2	
P3	3	0	2	9	0	2	
P4	2	1	1	2	2	2	
P5	0	0	2	4	3	3	
Total Allocated	**7**	**2**	**5**				

सगळ्यात आधी सर्व प्रोसेसची खालील फॉर्मुला वापरून नीड काढा

Need [i] = Max [i] - Allocation [i]

Need for P1: (7, 5, 3) - (0, 1, 0) = 7, 4, 3

Need for P2: (3, 2, 2) - (2, 0, 0) = 1, 2, 2

Need for P3: (9, 0, 2) - (3, 0, 2) = 6, 0, 0

Need for P4: (2, 2, 2) - (2, 1, 1) = 0, 1, 1

Need for P5: (4, 3, 3) - (0, 0, 2) = 4, 3, 1

Process	**Allocation** A	B	C	**Max** A	B	C	**Available** A	B	C	**Need** A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2				1	2	2
P3	3	0	2	9	0	2				6	0	0
P4	2	1	1	2	2	2				0	1	1
P5	0	0	2	4	3	3				4	3	1

बँकर्स अल्गोरिथम लागू करा

Available Resources of A, B and C are 3, 3, and 2. आता आपण प्रत्येक प्रोसेससाठी प्रत्येक प्रकारची संसाधन विनंती उपलब्ध आहे का ते तपासतो.

स्टेप 1: P1 प्रोसेस करिता: Need <= Available 7, 4, 3 <= 3, 3, 2 condition is false.

तर, आपण दुसरी प्रोसेस P2 तपासतो.

स्टेप 2: P2 प्रोसेस करिता: Need <= Available 1, 2, 2 <= 3, 3, 2 condition true

New available = available + Allocation

(3, 3, 2) + (2, 0, 0) => 5, 3, 2

Process	Allocation\ A	B	C	Max A	B	C	Available A	B	C	Need A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2	5	3	2	P2-Finish		
P3	3	0	2	9	0	2				6	0	0
P4	2	1	1	2	2	2				0	1	1
P5	0	0	2	4	3	3				4	3	1

त्याचप्रमाणे, आपण दुसरी प्रोसेस P3 तपासतो.

स्टेप 3: P3 प्रोसेस करिता: P3 Need <= Available 6, 0, 0 < = 5, 3, 2 condition is false.

त्याचप्रमाणे, आपण दुसरी प्रोसेस P4 तपासतो.

.स्टेप 4: P4 प्रोसेस करिता: P4 Need <= Available 0, 1, 1 <= 5, 3, 2 condition is true

New Available resource = Available + Allocation

5, 3, 2 + 2, 1, 1 => 7, 4, 3

Process	Allocation\ A	B	C	Max A	B	C	Available A	B	C	Need A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2	5	3	2	P2-Finish		
P3	3	0	2	9	0	2	7	4	3	6	0	0
P4	2	1	1	2	2	2				P4-Finish		
P5	0	0	2	4	3	3				4	3	1

त्याचप्रमाणे, आपण दुसरी प्रोसेस P5 तपासतो.

स्टेप 5: P5 प्रोसेस करिता: P5 Need <= Available 4, 3, 1 <= 7, 4, 3 condition is **true**

New available resource = Available + Allocation

7, 4, 3 + 0, 0, 2 => 7, 4, 5

Process	Allocation\ A	B	C	Max A	B	C	Available A	B	C	Need A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2	5	3	2	P2-Finish		
P3	3	0	2	9	0	2	7	4	3	6	0	0
P4	2	1	1	2	2	2	7	4	5	P4-Finish		
P5	0	0	2	4	3	3				P5-Finish		

आता, आपण पुन्हा एकदा P1 आणि P3 प्रोसेसेससाठी प्रत्येक प्रकारच्या संसाधन विनंतीचे परीक्षण करू..

स्टेप **6:** P1 प्रोसेस करिता: P1 Need <= Available 7, 4, 3 <= 7, 4, 5 condition is **true**

New Available Resource = Available + Allocation

7, 4, 5 + 0, 1, 0 => 7, 5, 5

Process	**Allocation\ A B C**	**Max A B C**	**Available A B C**	**Need A B C**
P1	0 1 0	7 5 3	3 3 2	P1-Finish
P2	2 0 0	3 2 2	5 3 2	P2-Finish
P3	3 0 2	9 0 2	7 4 3	6 0 0
P4	2 1 1	2 2 2	7 4 5	P4-Finish
P5	0 0 2	4 3 3	7 5 5	P5-Finish

आता, आपण शेवटची प्रोसेस P3 तपासू.

स्टेप **7:** P3 प्रोसेस करिता: P3 Need <= Available 6, 0, 0 <= 7, 5, 5 condition is true

New Available Resource = Available + Allocation

7, 5, 5 + 3, 0, 2 => 10, 5, 7

Process	**Allocation\ A B C**	**Max A B C**	**Available A B C**	**Need A B C**
P1	0 1 0	7 5 3	3 3 2	P1-Finish
P2	2 0 0	3 2 2	5 3 2	P2-Finish
P3	3 0 2	9 0 2	7 4 3	P3-Finish
P4	2 1 1	2 2 2	7 4 5	P4-Finish
P5	0 0 2	4 3 3	7 5 5	P5-Finish
सर्व प्रोसेस पूर्ण झाल्यानंतर उपलब्ध असलेला नवीन संसाधने सुरुवातीला उपलब्ध असलेल्या एकूण संसाधनांइतका असावा.			10 5 7	

उदाहरण 2: असे गृहीत धरा की 5 प्रोसेस आहेत, P0 ते P4, आणि 4 प्रकारचे संसाधने. टाइम T0 वर आपल्याकडे खालील सिस्टम स्थिती आहे. (Assume that there are 5 processes, P0 through P4, and 4 types of resources. At T0 we have the following system state)

Max Instances of Resource Type A = 3 (2 allocated + 1 Available)

Max Instances of Resource Type B = 17 (12 allocated + 5 Available)

Max Instances of Resource Type C = 16 (14 allocated + 2 Available)

Max Instances of Resource Type D = 12 (12 allocated + 0 Available)

PID	Allocation				Maximum Recourses Required				Not Allocated (Available)Resources			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	0	1	1	0	0	2	1	0	1	5	2	0
P1	1	2	3	1	1	6	5	2				
P2	1	3	6	5	2	3	6	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				
Total	2	12	14	12								

नीड मॅट्रिक्स (Need Matrix) खाली दिल्या प्रमाणे तयार करा

PID	Allocation				Maximum Recourses Required				Need = Max-Allocated			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	0	1	1	0	0	2	1	0	0	1	0	0
P1	1	2	3	1	1	6	5	2	0	4	2	1
P2	1	3	6	5	2	3	6	6	1	0	0	1
P3	0	6	3	2	0	6	5	2	0	0	2	0
P4	0	0	1	4	0	6	5	6	0	6	4	2
Total	2	12	14	12								

प्रोसेस करिता P0

Need of P0(0,1,0,0)<= Avail(1,5,2,0) == TRUE नंतर प्रोसेस P0 पूर्ण करा

PID	Allocation				Maximum Recourses Required				Not Allocated (Available)Resources				Status	Finish.
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**		
P0	0	1	1	0	0	2	1	0	1	5	2	0	Completed	1
P1	1	2	3	1	1	6	5	2						
P2	1	3	6	5	2	3	6	6						
P3	0	6	3	2	0	6	5	2						
P4	0	0	1	4	0	6	5	6						
Total	2	12	14	12										

आता उपलब्ध संसाधने अपडेट करा. Avail(A,B,C,D)+Allocated(A,B,C,D) of P0

Updated Avail(1,6,3,0) =Avail(1,5,2,0) + Allocated(0,1,1,0)

PID	Allocation				Maximum Recourses Required				Not Allocated (Available)Resources				Status	Finish.
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**		
P0	0	1	1	0	0	2	1	0	1	5	2	0	Completed	1
P1	1	2	3	1	1	6	5	2	1	6	3	0	Pending	
P2	1	3	6	5	2	3	6	6					Pending	
P3	0	6	3	2	0	6	5	2						
P4	0	0	1	4	0	6	5	6						
Total	2	12	14	12										

P1 प्रोसेस चेक करा

Need of P1(0,4,2,1)<=Avail(1,6,3,0) == FALSE, म्हणून P1 एक्झिक्युट करता येत नाही.

P2 प्रोसेस चेक करा

Need of P2(1,0,0,1)<=Avail(1,6,3,0) == FALSE, म्हणून P2 एक्झिक्युट करता येत नाही.

P3 प्रोसेस चेक करा

Need of P3(0,0,2,0)<=Avail(1,6,3,0) == TRUE, म्हणून P3 एक्झिक्युट करता येते

आता उपलब्ध संसाधने अपडेट करा, Avail(A,B,C,D)+Allocated(A,B,C,D) of P3

Updated Avail(1,12,6,2) =Avail(1,6,3,0) + Allocated(0,6,3,2)

PID	Allocation				Maximum Recourses Required				Not Allocated (Available)Resources				Status	Finish.
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**		
P0	0	1	1	0	0	2	1	0	1	5	2	0	Completed	1
P1	1	2	3	1	1	6	5	2	1	6	3	0	Pending	
P2	1	3	6	5	2	3	6	6	1	12	6	2	Pending	
P3	0	6	3	2	0	6	5	2	1	12	7	6	Completed	2
P4	0	0	1	4	0	6	5	6					Completed	3
Total	2	12	14	12										

Need of P4(0,6,4,2)<=Avail(1,12,6,2) == TRUE, म्हणून P4 एक्झिक्युट करता येते.

आता उपलब्ध संसाधने अपडेट करा, Avail(A,B,C,D)+Allocated(A,B,C,D) of P4

Updated Avail(1,12,7,6) =Avail(1,12,6,2) + Allocated(0,0,1,4)

P1 प्रोसेस पुन्हा चेक करा

Need of P1(0,4,2,1)<=Avail(1,12,7,6) == TRUE, म्हणून P1 एक्झिक्युट करता येते.

आता उपलब्ध संसाधने अपडेट करा, Avail(A,B,C,D)+Allocated(A,B,C,D) of P1

Updated Avail(2,14,10,7) =Avail(1,12,7,6) + Allocated(1,2,3,1)

PID	Allocation				Maximum Recourses Required				Not Allocated (Available)Resources				Status	Finish.
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**		
P0	0	1	1	0	0	2	1	0	1	5	2	0	Completed	1
P1	1	2	3	1	1	6	5	2	1	6	3	0	Completed	4
P2	1	3	6	5	2	3	6	6	1	12	6	2	Pending	
P3	0	6	3	2	0	6	5	2	1	12	7	6	Completed	2
P4	0	0	1	4	0	6	5	6	2	14	10	7	Completed	3
Total	2	12	14	12										

P2 प्रोसेस पुन्हा चेक करा

Need of P2(1,0,0,1)<=Avail(2,14,10,7) will be TRUE, म्हणून P2 एक्झिक्युट करता येते.

आता उपलब्ध संसाधने अपडेट करा , Avail(A,B,C,D)+Allocated(A,B,C,D) of P1

Updated Avail(3,17,16,12) =Avail(2,14,10,7) + Allocated(1,3,6,5)

PID	Allocation				Maximum Recourses Required				Not Allocated (Available)Resources				Status	Finish.
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**		
P0	0	1	1	0	0	2	1	0	1	5	2	0	Completed	1
P1	1	2	3	1	1	6	5	2	1	6	3	0	Completed	4
P2	1	3	6	5	2	3	6	6	1	12	6	2	Completed	5
P3	0	6	3	2	0	6	5	2	1	12	7	6	Completed	2
P4	0	0	1	4	0	6	5	6	2	14	10	7	Completed	3
Total	2	12	14	12					3	17	16	12		

सिस्टम सुरक्षित स्थितीत आहे आणि प्रोसेस खालील क्रमाने एक्झिक्युट केल्या जातील:

P0→P3→P→4→P1→P2

References:

1. Operating System: A Concept-Based Approach by Dhananjay M. Dhamdhere, McGraw Hill Education 3rd edition, ISBN: 978-1259005589
2. Operating Systems : Internals and Design Principles by William Stallings, Pearson Education 9th Edition, ISBN: 978-9352866717
3. Linux The Complete Reference by Richard Petersen, McGraw Hill, 6th edition, ISBN: 978-0071492478
4. Linux command line and shell scripting by Richard Blum, Wiley India, ISBN: 978-1118983843
5. Operating System Concepts by Abraham Silberschatz and James Peterson, Wiley India, ISBN: 9781119454083

Websites:

1. https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm
2. https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
3. https://www.scaler.com/topics/operating-system/deadlock-in-os/
4. https://www.javatpoint.com/os-deadlocks-introduction
5. https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/
6. https://www.tutorialspoint.com/process-deadlocks-in-operating-system
7. https://www.studytonight.com/operating-system/deadlocks

E-learning material (Video lectures)

1. https://www.youtube.com/watch?v=bWHFY8-rL5I –Scheduling criterion
2. https://www.youtube.com/watch?v=eoGUQ5sl0vw –Scheduling criterion
3. https://www.youtube.com/watch?v=zFnrUVqtiOY –Scheduling Types
4. https://www.youtube.com/watch?v=MZdVAVMgNpA – FCFS Scheduling algorithm
5. https://www.youtube.com/watch?v=VCIVXPoiLpU – SJF Scheduling algorithm
6. https://www.youtube.com/watch?v=hoN7_VMzw_g –SRTN Scheduling algorithm
7. https://www.youtube.com/watch?v=TxjIlNYRZ5M – RR Scheduling Algorithm
8. https://www.youtube.com/watch?v=rsDGfFxSgiY – Priority Scheduling Algorithm
9. https://www.youtube.com/watch?v=hBPYP0ZEvS8 – Multi level Queue Scheduling
10. https://www.youtube.com/watch?v=rWFH6PLOIEI – Deadlock
11. https://www.youtube.com/watch?v=01DiVzZbRjY – Starvation and Aging in OS
12. https://www.youtube.com/watch?v=7gMLNiEz3nw – Bankers's Algorithm

युनिट-4
मेमरी मॅनेजमेंट
(Memory Management)

विषय निष्पत्ती (Course Outcome):

CO4: ऑपरेटिंग सिस्टमद्वारे वापरल्या जाणाऱ्या मेमरी मॅनेजमेंट तंत्रांचे विश्लेषण करा.

घटक निष्पत्ती (Theory Learning Outcome-TLO):

1. फिक्स्ड आणि व्हेरिएबल मेमरी पार्टीशनची तुलना करा.
2. बिट मॅप आणि लिंक्ड लिस्ट तंत्रात फरक करा.
3. विविध पार्टीशन अल्गोरिथमचे कार्य स्पष्ट करा.
4. दिलेल्या पेज रेफरन्स स्ट्रिंगसाठी पेज फॉल्टची गणना करा.

4.1 मूलभूत मेमरी मॅनेजमेंट (Basic memory management)

मेमरी मॅनेजमेंट ही ऑपरेटिंग सिस्टमची कार्यक्षमता आहे जी प्रायमरी मेमरी हाताळते किंवा व्यवस्थापित करते आणि एक्झिक्युशन दरम्यान मुख्य मेमरी आणि डिस्क दरम्यान प्रोसेसेस पुढे-मागे हलवते. मेमरी मॅनेजमेंट प्रत्येक मेमरी लोकेशनचा मागोवा ठेवते, ते कोणत्याही प्रोसेस साठी वाटप केलेले आहे किंवा ते रिकामे आहे याची पर्वा न करता.

1. ते प्रोसेसेसना किती मेमरी अलोकेट करायची आहे ते तपासते.
2. कोणत्या प्रोसेसला कोणत्या वेळी मेमरी मिळेल हे ते ठरवते.
3. काही मेमरी केव्हा मोकळी होते किंवा अनअलोकेटेड होते याचा मागोवा घेते आणि त्यानुसार स्टेटस अपडेट करते.

Fig. 4.1 मध्ये दाखवल्याप्रमाणे, मेमरी मॅनेजमेंटच्या दोन पद्धती आहेत: कॉन्टिग्युअस (Contiguous) आणि नॉन-कॉन्टिग्युअस (Non-Contiguous).

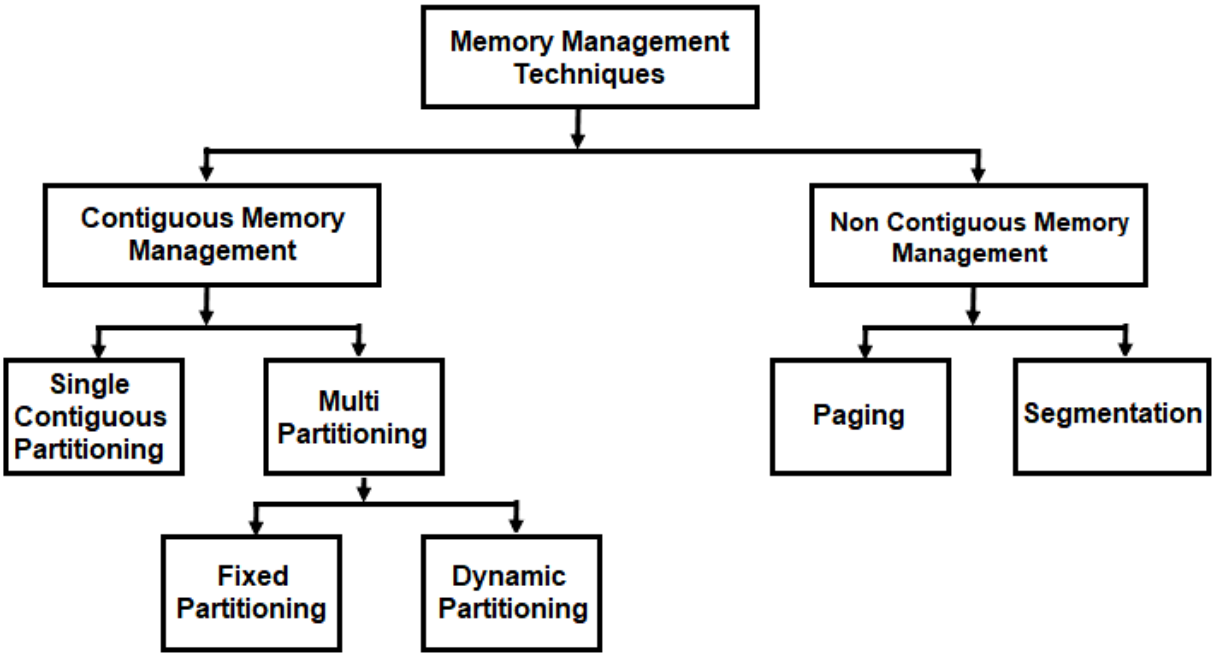


Fig.4.1: मेमरी मॅनेजमेंट तंत्रे (Memory Management Techniques)

4.2 कॉन्टिग्युअस मेमरी मॅनेजमेंट टेक्निक (Contiguous memory management Technique)

कॉन्टिग्युअस मेमरी मॅनेजमेंट टेक्निकमध्ये, प्रत्येक प्रोग्राम स्टोरेज लोकेशन्सचा एकच कॉन्टिग्युअस ब्लॉक व्यापतो, म्हणजेच, सलग ॲड्रेस सह (Consecutive addresses) मेमरी लोकेशन्सचा संच. कॉन्टिग्युअस टेक्निक दोन भागामध्ये विभागले जाऊ शकते:

1. सिंगल कॉन्टिग्युअस मेमरी मॅनेजमेंट तंत्र (Single contiguous memory management Technique)
2. मल्टिपल पार्टीशनिंग (Multiple Partitioning)

1. सिंगल कॉन्टिग्युअस मेमरी मॅनेजमेंट तंत्र (Single contiguous memory management Technique)

सिंगल कॉन्टिग्युअस मेमरी मॅनेजमेंट टेक्निक ही संगणक प्रणालीच्या सुरुवातीच्या पिढीमध्ये वापरली जाणारी सर्वात सोपी मेमरी मॅनेजमेंट टेक्निक आहे. या टेक्निकमध्ये, मुख्य मेमरी दोन कॉन्टिग्युअस एरिया किंवा पार्टीशनमध्ये विभागली जाते. ऑपरेटिंग सिस्टम कायमस्वरूपी एका पार्टीशनमध्ये, सामान्यतः लोवर मेमरीमध्ये असते आणि यूजर्स प्रोसेस दुसऱ्या पार्टीशनमध्ये लोड केली जाते.

फायदे (**Advantages)**:

1. इम्प्लिमेंट करण्यास सोपे आहे.
2. मॅनेज आणि डिझाइन करणे सोपे आहे .
3. सिंगल कॉन्टिग्युअस मेमरी मॅनेजमेंट तंत्रात, एकदा प्रोसेस लोड झाल्यानंतर, तिला पूर्ण प्रोसेसरचा वेळ दिला जातो आणि दुसरा कोणताही प्रोसेसर त्यात व्यत्यय आणत नाही.

तोटे (**Disadvantages)**:

1. न वापरल्या मेमरीमुळे, मेमरी स्पेसचा अपव्यय होतो कारण प्रोसेस मध्ये सर्व उपलब्ध मेमरी स्पेस वापरण्याची शक्यता कमी असते.
2. सीपीयू निष्क्रिय राहतो, डिस्क बायनरी इमेज मुख्य मेमरीमध्ये लोड होण्याची वाट पाहत असतो.
3. जर प्रोग्राम संपूर्ण उपलब्ध मेमरी स्पेसमध्ये स्टोअर करण्यासाठी खूप मोठा असेल तर तो एक्झिक्युट करता येत नाही.
4. ते मल्टीप्रोग्रामिंगला समर्थन देत नाही, म्हणजेच ते एकाच वेळी अनेक प्रोग्राम हाताळू शकत नाही.

2. मल्टिपल पार्टीशनिंग (Multiple Partitioning)

सिंगल कॉन्टिग्युअस मेमरी मॅनेजमेंट तंत्र अकार्यक्षम आहे कारण ते संगणकांना एका वेळी फक्त एकच प्रोग्राम एक्झिक्युट करण्यास मर्यादित करते ज्यामुळे मेमरी स्पेस आणि सीपीयू वेळेचा अपव्यय होतो. अकार्यक्षम सीपीयू वापराची समस्या मल्टीप्रोग्रामिंग वापरून दूर केली जाऊ शकते जी एकापेक्षा जास्त प्रोग्राम एकाच वेळी रन करण्यास अनुमती देते. दोन प्रोसेसेसमध्ये स्विच करण्यासाठी, ऑपरेटिंग सिस्टमला दोन्ही प्रोसेस मुख्य मेमरीमध्ये लोड कराव्या लागतात. मुख्य मेमरीमध्ये अनेक प्रोसेस लोड करण्यासाठी ऑपरेटिंग सिस्टमला उपलब्ध मेन मेमरीचे अनेक पार्टीशनमध्ये विभाजन करावे लागते. अशा प्रकारे अनेक प्रक्रिया एकाच वेळी मुख्य मेमरीमध्ये राहू शकतात. मल्टी पार्टीशन तंत्र दोन प्रकारचे असू शकते.

a. फिक्स्ड (स्टॅटिक) पार्टीशनिंग (Fixed (Static) Partitioning)
b. व्हेरिएबल (डायनॅमिक) पार्टीशनिंग (Variable (Dynamic) Partitioning)

a. फिक्स्ड (स्टॅटिक) पार्टीशनिंग (Fixed (Static) Partitioning)

मुख्य मेमरीमध्ये एकापेक्षा जास्त प्रक्रिया ठेवण्यासाठी वापरली जाणारी ही सर्वात जुनी आणि सोपी पद्धत आहे. या पार्टीशनिंगमध्ये , RAM मधील पार्टीशनची संख्या (नॉन-ओव्हरलॅपिंग) निश्चित असते परंतु प्रत्येक पार्टीशनचा आकार समान असू शकतो किंवा नसू शकतो. हे कॉन्टिग्युअस वाटप असल्याने, कोणतेही स्पॅनिंग परवानगी नाही. येथे पार्टिशन्स एक्झिक्युशनपूर्वी किंवा सिस्टम कॉन्फिगरेशन दरम्यान केली जातात. मुख्य मेमरी एका फिक्स्ड पार्टीशन मेमरी मॅनेजमेंट तंत्र किंवा स्टॅटिक पार्टीशन अनेक फिक्स्ड-आकाराच्या पार्टीशनमध्ये विभागली जाते. हे पार्टिशन्स समान आकाराचे किंवा वेगवेगळ्या आकाराचे असू शकतात. प्रत्येक विभाजन एकच प्रोसेस स्टोअर करू शकते. पार्टिशन्सची संख्या मल्टीप्रोग्रामिंगची डिग्री ठरवते, म्हणजेच मेमरीमधील प्रोसेसेसची कमाल संख्या. हे पार्टीशन सिस्टम जनरेशनच्या वेळी केले जातात आणि Fig. 4.2 मध्ये दाखवल्याप्रमाणे त्यानंतर फिक्स्ड राहतात.

फिक्स्ड पार्टिशनिंगचे फायदे:

1. इम्प्लिमेंट करण्यास सोपे
2. कमी ओएस ओव्हरहेड

फिक्स्ड पार्टिशनिंगचे तोटे:

1. इंटर्नल फ्रॅगमेंटेशन
2. एक्सटर्नल फ्रॅगमेंटेशन

3. प्रोसेसचा आकार मर्यादित करते
4. मल्टीप्रोग्रामिंगच्या डिग्रीवर मर्यादा
5. सिस्टम जनरेशनच्या वेळी पार्टिशन्सची संख्या ठरविली जाते.

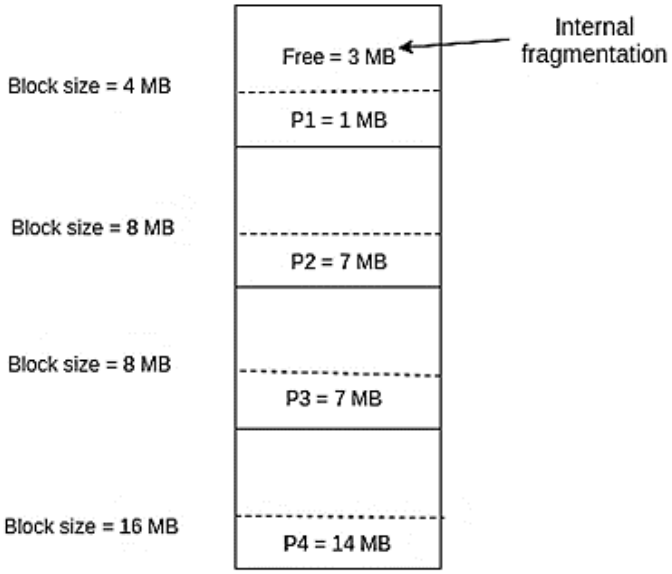


Fig.4.2: फिक्स्ड (स्टॅटिक) पार्टीशनिंग (Fixed (Static) Partitioning)

फिक्स्ड पार्टीशन पद्धतीमध्ये मेमरी वाटप (Memory Allocation in Fixed Partition Method)

सुरुवातीला, सर्व मेमरी यूजर्सच्या प्रोसेससाठी उपलब्ध असते आणि उपलब्ध मेमरीचा एक मोठा ब्लॉक मानला जातो. ही उपलब्ध मेमरी "**होल**" (Memory Hole) म्हणून ओळखली जाते. जेव्हा प्रोसेस येते आणि मेमरीची आवश्यकता असते, तेव्हा ही प्रोसेस स्टोअर करण्यासाठी पुरेसे मोठे असलेल्या होल शोधतो. जर आवश्यकता पूर्ण झाली तर प्रोसेससाठी मेमरीचे वाटप करतो, अन्यथा भविष्यातील मेमोरीच्या विनंत्या पूर्ण करण्यासाठी उर्वरित उपलब्ध ठेवते. मेमरीचे वाटप करताना कधीकधी-डायनॅमिक स्टोरेज वाटप समस्या उद्भवतात, ज्यामुळे फ्री होलच्या सूचीमधून आकार एनची विनंती कशी पूर्ण करावी याबद्दल विचार करते. या समस्येवर काही उपाय आहेत आणि सर्वात जास्त वापरल्या जाणाऱ्या चार वाटप तंत्रे आहेत

1. फर्स्ट फिट (First Fit)
2. बेस्ट फिट (Best Fit)
3. वर्स्ट फिट (Worst Fit)

1. फर्स्ट फिट (First Fit)

जेव्हा जेव्हा एखादी प्रोसेस (P1) मेमरी अलोकेशन रिक्वेस्टसह येते तेव्हा खालील गोष्टी घडतात -

1. OS पहिल्या इंडेक्समधून उपलब्ध मेमरी ब्लॉक्स अनुक्रमे शोधते
2. प्रोसेस सामावून घेण्यासाठी पुरेसा मोठा असलेला पहिला मेमरी ब्लॉक नियुक्त करते

जेव्हा जेव्हा जेव्हा नवीन प्रक्रिया P2 येते तेव्हा ते तेच काम करते म्हणजेच पहिल्या इंडेक्समधून पुन्हा शोधते.

उदाहरण:

उपलब्ध मेमरी ब्लॉक्स आहेत: {100, 50, 30, 120, 35}

प्रोसेस P1, आकार: 20

- OS सुरुवातीपासून सिक्वेन्शिअल मेमरी शोधते
- ब्लॉक 1 फिट बसतो, म्हणून P1 ब्लॉक 1 ला अलोकेट केला जातो

प्रोसेस P2, आकार: 60

- OS ब्लॉक 1 पासून पुन्हा सिक्वेन्शिअल मेमरी शोधते
- ब्लॉक 1 अनुपलब्ध आहे, ब्लॉक 2 आणि 3 बसू शकत नाही

- ब्लॉक 4 फिट बसतो, P2 ब्लॉक 4 ला अलोकेट केला जातो

प्रोसेस P3, आकार: 70

- OS ब्लॉक 1 मधून पुन्हा सिक्वेन्शिअल मेमरी शोधते
- ब्लॉक 1 अनुपलब्ध आहे, ब्लॉक 2, 3 फिट होऊ शकत नाही. ब्लॉक 4 अनुपलब्ध, ब्लॉक 5 फिट होऊ शकत नाही आणि P3 अनअलोकेट राहते

त्याचप्रमाणे, Fig 4.3 मध्ये दर्शविल्यानुसार P4 ब्लॉक 2 वर अलोकेट केले आहे

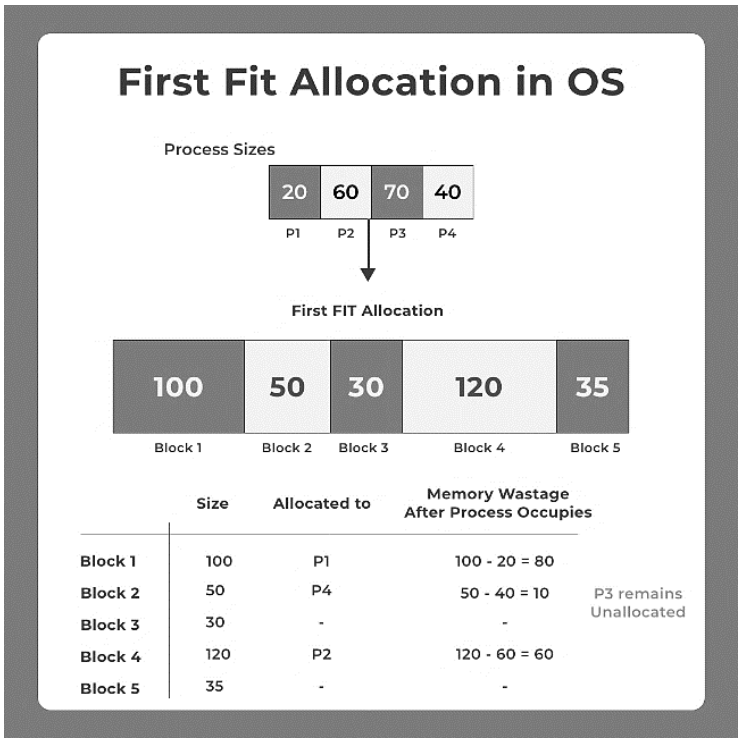


Fig.4.3: फर्स्ट फिट (First Fit)

फायदे (Advantages)

1. इम्प्लिमेंट करणे सोपे आहे
2. २. इतर पद्धतींच्या तुलनेत प्रोसेसेसचे वाटप करण्यासाठी अल्गोरिथम जलद असल्याने ऑपरेटिंग सिस्टम प्रोसेसचे वाटप जलद करू शकते.

तोटे (Disadvantages)

1. मोठ्या प्रमाणात इंटर्नल फ्रॅगमेंटेशन होते
2. खराब अल्गोरिथममुळे काही प्रोसेसेसचे वाटप रद्द होण्याची उच्च शक्यता असते.
3. नेक्स्ट फिटच्या तुलनेत जास्त ओव्हरहेड असते.

2. बेस्ट फिट (Best Fit)

बेस्ट फिट मेमरी अलोकेशन टेक्निकमध्ये, ऑपरेटिंग सिस्टम असे मेमरी ब्लॉक्स शोधते ज्यात प्रोसेसला सामावून घेऊ शकतात आणि कमीतकमी मेमरी वाया घालवतात.

उदाहरणः

जर तुम्ही खाली दिलेले Fig. 4.4 पाहिले तर तुम्हाला दिसेल की प्रोसेसचा आकार 40 आहे. तर ब्लॉक 1, 2 आणि 4 मध्ये प्रोसेस सामावून घेता येते. पण ब्लॉक 2 निवडला जातो कारण तो सर्वात कमी मेमरी वाया घालवतो.

फायदे (Advantages)

1. ते प्रोसेससाठी वाटप केले जाते आणि ही टेक्निक सर्वोत्तम मानली जातात कारण त्यामुळे सर्वात ऑप्टिमाइझ्ड मेमरी वाटप होते.
2. इंटर्नल फ्रॅगमेंटेशन देखील कमी होते.

तोटे (Disadvantages)

1. तथापि, बेस्ट-फिट मेमरी वाटप शोधणे वेळखाऊ असू शकते.

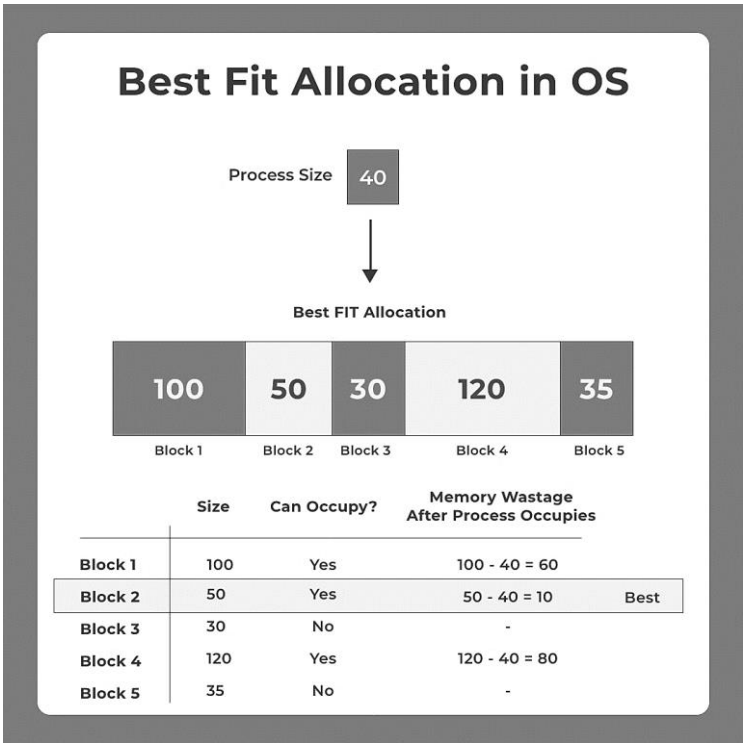


Fig.4.4: बेस्ट फिट (Best Fit)

3. वर्स्ट फिट (Worst Fit)

कोणत्याही दिलेल्या प्रोसेस Pn साठी, वर्स्ट फिट खालील प्रकारे कार्य करते. अल्गोरिथम पहिल्या मेमरी ब्लॉकपासून सिक्वेन्शिअल शोधतात आणि खालील अट पूर्ण करणाऱ्या मेमरी ब्लॉकचा शोध घेतात:

1. प्रोसेसचा आकार सामावून घेऊ शकतात
2. Fig. 4.5 मध्ये दाखवल्याप्रमाणे दिलेल्या मेमरी ब्लॉकला प्रोसेस वाटप केल्यानंतर सर्वात मोठी वाया जाणारी जागा (फ्रॅगमेंटेशन) सोडते.

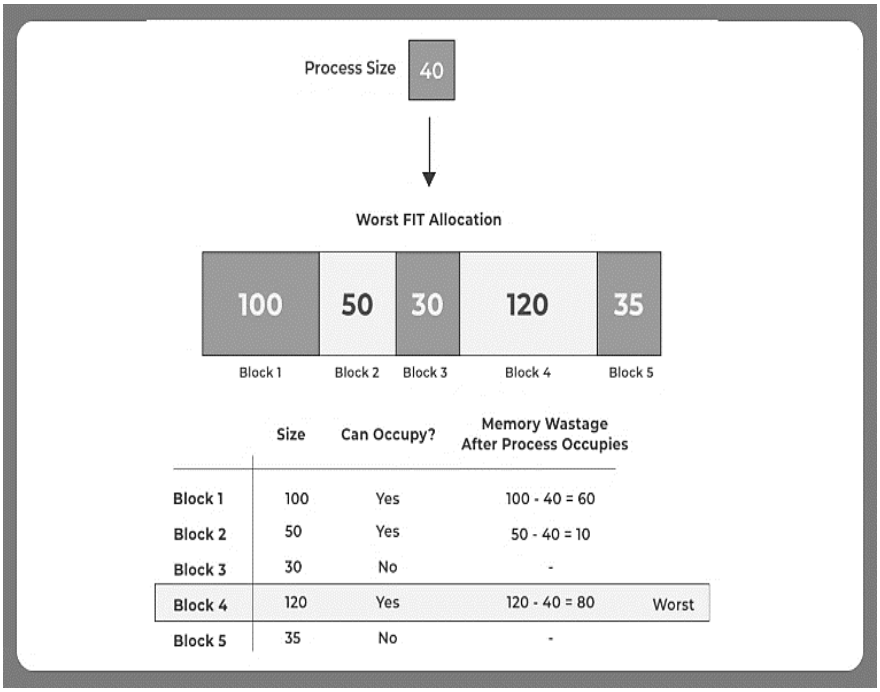


Fig.4.5: वर्स्ट फिट (Worst Fit)

b. व्हेरिएबल (डायनॅमिक) पार्टीशनिंग (Variable (Dynamic) Partitioning)

व्हेरिअबल पार्टिशनिंग हे कॉन्टिग्युअस अलोकेशन तंत्राचा एक भाग आहे आणि फिक्स्ड पार्टिशनिंगमध्ये येणाऱ्या समस्येचे निराकरण करण्यासाठी वापरले जाते. फिक्स्ड पार्टिशनिंगच्या विपरीत, एक्झिक्युशनपूर्वी किंवा सिस्टम कॉन्फिगरेशन दरम्यान पार्टिशन बनवले जात नाहीत. व्हेरिअबल पार्टिशनिंगशी संबंधित विविध वैशिष्ट्ये-

1. सुरुवातीला रॅम रिकामी असते आणि सिस्टम कॉन्फिगरेशन दरम्यान पार्टीशन करण्याऐवजी प्रोसेसेसच्या गरजेनुसार रन-टाइम दरम्यान पार्टिशन्स केली जातात.
2. पार्टीशनचा आकार येणाऱ्या प्रोसेस इतकाच असतो.
3. प्रोसेसच्या गरजेनुसार पार्टीशनचा आकार बदलतो जेणेकरून इंटर्नल फ्रॅगमेंटेशन टाळता येईल आणि RAM चा कार्यक्षम वापर सुनिश्चित करता येईल.
4. RAM मधील पार्टिशन्सची संख्या फिक्स्ड नसते आणि ती Fig. 4.6 मध्ये दाखवल्याप्रमाणे येणाऱ्या प्रोसेसेसची संख्या आणि मेन मेमरीच्या आकारावर अवलंबून असते.

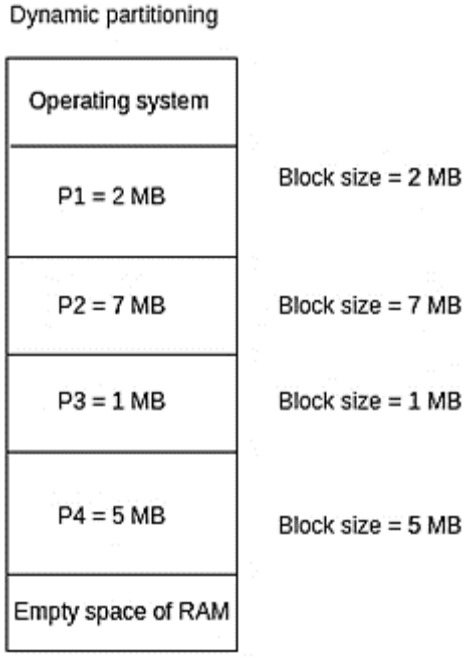


Fig. 4.6: व्हेरिएबल (डायनॅमिक) पार्टीशनिंग (Variable (Dynamic) Partitioning)

4.3 ऑपरेटिंग सिस्टममध्ये फ्री स्पेस मॅनेजमेंट टेक्निक्स (Free space management techniques in Operating System)

फाइल्स तयार केल्यावर त्यांना जागा वाटण्यासाठी सिस्टम फ्री डिस्क ब्लॉक्सचा ट्रॅक ठेवते. तसेच, फाइल्स डिलीट केल्यापासून मोकळी झालेली जागा पुन्हा वापरण्यासाठी, फ्री स्पेस मॅनेजमेंट अत्यंत महत्वाचे बनते. सिस्टम फ्री स्पेस लिस्ट ठेवते जी काही फाइल किंवा डायरेक्टरीमध्ये वाटप न केलेल्या डिस्क ब्लॉक्सचा ट्रॅक ठेवते. फ्री स्पेस लिस्ट प्रामुख्याने खालीलप्रमाणे लागू केली जाऊ शकते:

4.3.1 बिटमॅप किंवा बिट वेक्टर (Bitmap or Bit vector)

बिटमॅप किंवा बिट वेक्टर ही बिट्सची मालिका किंवा संग्रह आहे जिथे प्रत्येक बिट Fig. 4.7 मध्ये दाखवल्याप्रमाणे डिस्क ब्लॉकशी समंधित असते. बिट दोन मूल्ये घेऊ शकतो: 0 आणि 1: 0 ब्लॉक वाटप केल्याचे दर्शविते आणि 1 एक फ्री ब्लॉक दर्शविते. Fig. 4.7 मध्ये डिस्कवरील डिस्क ब्लॉक्सचे दिलेले उदाहरण (जिथे ग्रे ब्लॉक वाटप केले आहेत) 16 बिट्सच्या बिटमॅपद्वारे दर्शविले जाऊ शकते: 0000111000000110.

फायदे (Advantages)

1. सोपी आणि समजण्यास सोपी.
2. कमी मेमरी वापरते.
3. फ्री स्पेस शोधण्यास कार्यक्षम आहे.

तोटे (Disadvantages)

1. ऑपरेटिंग सिस्टम सर्व ब्लॉक्समधून जाते जोपर्यंत तिला एक फ्री ब्लॉक सापडत नाही. (ज्याचा बिट '0' आहे असा ब्लॉक).
2. डिस्कचा आकार मोठा असताना ते कार्यक्षम नसते.

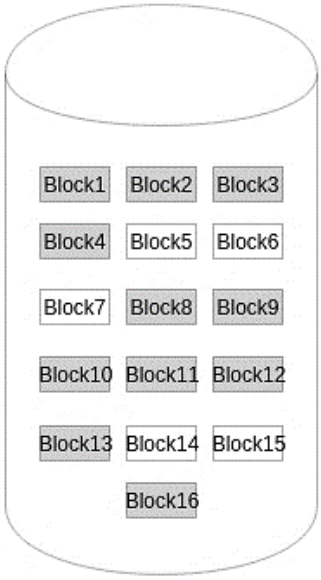


Fig. 4.7: बिटमॅप किंवा बिट वेक्टर (Bitmap or Bit vector)

4.3.2 लिंक्ड लिस्ट (Linked List)

या दृष्टिकोनात, फ्री डिस्क ब्लॉक्स एकमेकांशी जोडलेले असतात म्हणजेच फ्री ब्लॉकमध्ये Fig. 4.8 मध्ये दाखवल्याप्रमाणे पुढील फ्री ब्लॉककडे जाण्यासाठी पॉइंटर असतो. पहिल्या डिस्क ब्लॉकचा ब्लॉक नंबर डिस्कवर वेगळ्या ठिकाणी संग्रहित केला जातो आणि मेमरीमध्ये देखील कॅचे (Cache) केला जातो.

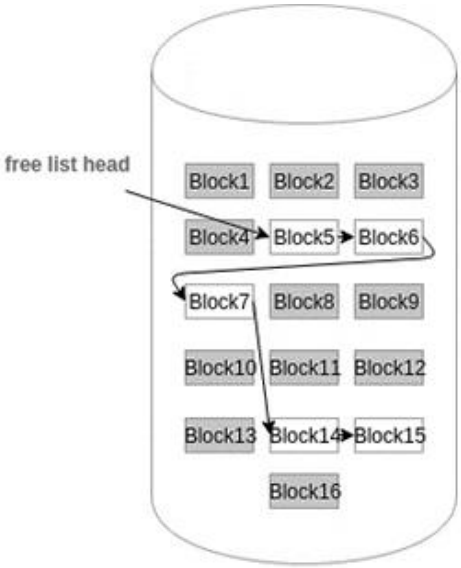


Fig. 4.8: लिंक्ड लिस्ट (Linked List)

Fig. 4 .8 मध्ये, फ्री स्पेस लिस्ट हेड ब्लॉक 5-कडे पॉईंट करते आणि ब्लॉक 5-ब्लॉक 6ला पॉईंट करते, पुढील फ्री ब्लॉक इत्यादींकडे पॉईंट करते. शेवटच्या फ्री ब्लॉकमध्ये फ्री स्पेस लिस्टचा शेवट दर्शविणारा एक नल पॉइंटर असेल. या पद्धतीचा एक तोटा म्हणजे फ्री स्पेस लिस्ट ट्रॅव्हर्सलसाठी आवश्यक असलेला I/O.

फायदे (Advantages)

1. या पद्धतीत, उपलब्ध जागेचा कार्यक्षमतेने वापर केला जातो.
2. लिंक्ड लिस्टचा आकारावर मर्यादा नसल्यामुळे, नवीन फ्री स्पेस सहजपणे जोडता येते.

तोटे (Disadvantages)

1. या पद्धतीत, पॉइंटर राखण्याचा ओव्हरहेड दिसून येतो.
2. जेव्हा आपल्याला मेमरीच्या प्रत्येक ब्लॉकपर्यंत पोहोचण्याची आवश्यकता असते तेव्हा लिंक्ड लिस्ट कार्यक्षम नसते.

4.4 स्वॅपिंग (Swapping)

मल्टीप्रोग्रामिंगमध्ये CPU चा वापर वाढवण्यासाठी, स्वॅपिंग म्हणून ओळखल्या जाणाऱ्या मेमरी मॅनेजमेंट टेक्निकचा वापर केला जाऊ शकतो. स्वॅपिंग म्हणजे प्रोसेस मेमरीमध्ये आणण्याची आणि नंतर ती काही काळ रन झाल्यानंतर तात्पुरती डिस्कवर कॉपी करण्याची प्रक्रिया आहे. ऑपरेटिंग सिस्टममध्ये स्वॅपिंगचा उद्देश हार्ड डिस्कवरील डेटा ऍक्सेस करणे आणि तो RAM मध्ये आणणे आहे जेणेकरून अप्लिकेशन प्रोग्राम त्याचा वापर करू शकतील. हे लक्षात ठेवणे महत्त्वाचे आहे की RAM मध्ये डेटा उपलब्ध नसतानाच स्वॅपिंग वापरले जाते. स्वॅपिंग प्रक्रिया सिस्टमची कार्यक्षमता कमी करते, परंतु ती मोठ्या आणि अनेक प्रोसेस एकाच वेळी चालविण्यास अनुमती देते. यामुळे, स्वॅपिंगला मेमरी कॉम्पॅक्शन असेही म्हणतात. सीपीयू शेड्युलर कोणत्या प्रोसेस स्वॅप इन करायच्या आणि कोणत्या स्वॅप आउट करायच्या हे ठरवतो. Fig. 4.9 ऑपरेटिंग सिस्टममधील स्वॅपिंग प्रक्रिया दर्शवते. स्वॅपिंग दोन संकल्पनांमध्ये विभागले गेले आहे: स्वॅप-इन आणि स्वॅप-आउट.

- स्वॅप-आउट (Swap-out) ही रॅम वरून हार्ड डिस्कवर प्रोसेस हलवण्याची एक पद्धत आहे.
- स्वॅप-इन (Swap-in) ही हार्ड डिस्कवरून मुख्य मेमरी किंवा रॅममध्ये प्रोग्राम ट्रान्स्फर करण्याची एक पद्धत आहे.

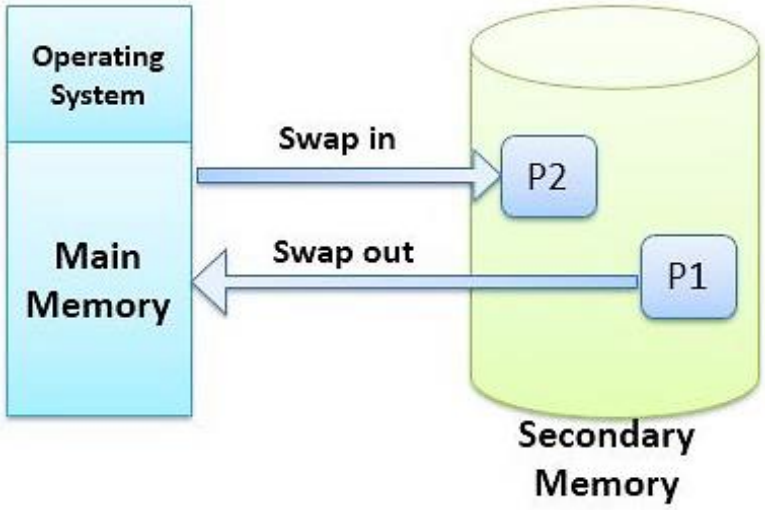


Fig.4.9: स्वॅपिंग (Swapping)

4.5 कॉम्पॅक्शन (Compaction)

मेमरी कॉम्पॅक्शन ही एक टेक्निक आहे जी फ्रॅगमेंटच्या स्वरूपात असलेल्या संपूर्ण फ्री मेमरी ब्लॉकला फ्री मेमरीच्या एका मोठ्या फ्रॅगमेंटमध्ये गोळा करते, ज्याचा वापर Fig. 4.10 मध्ये दाखवल्याप्रमाणे इतर प्रोसेस रन करण्यासाठी साठी केला जाऊ शकतो. हे सर्व प्रोसेस मेमरीच्या एका टोकाकडे आणि सर्व उपलब्ध मोकळी जागा मेमरीच्या दुसऱ्या टोकाकडे हलवून असे करते जेणेकरून ते कॉन्टिग्युअस होईल. कॉम्पॅक्शन करणे नेहमीच सोपे नसते. कॉम्पॅक्शन फक्त तेव्हाच करता येते जेव्हा रिलोकेशन डायनॅमिक असते आणि एक्झिक्युशनच्या वेळी केले जाते. कॉम्पॅक्शन तेव्हाच करता येते जेव्हा रिलोकेशन स्थिर असते आणि लोड टाइमवर किंवा असेंब्लीच्या वेळी केले जाते.

कॉम्पॅक्शनपूर्वी: कॉम्पॅक्शनपूर्वी, मुख्य मेमरीमध्ये व्यापलेल्या जागेमध्ये काही मोकळी जागा असते आणि ही स्थिती एक्सटर्नल फ्रॅगमेंटेशन म्हणून ओळखली जाते. व्यापलेल्या स्पेसमधील कमी फ्री स्पेस असल्याने, मोठ्या प्रोसेस त्यामध्ये लोड केल्या जाऊ शकत नाहीत.

कॉम्पॅक्शननंतर: कॉम्पॅक्शननंतर, सर्व व्यापलेली स्पेस वर हलवली जाते आणि खाली फ्री स्पेस असते. यामुळे स्पेस कॉन्टिग्युअस होते आणि एक्सटर्नल फ्रॅगमेंटेशन दूर होते. मोठ्या मेमरी आवश्यकता असलेल्या प्रोसेसेस आता मेन मेमरीमध्ये लोड केल्या जाऊ शकतात.

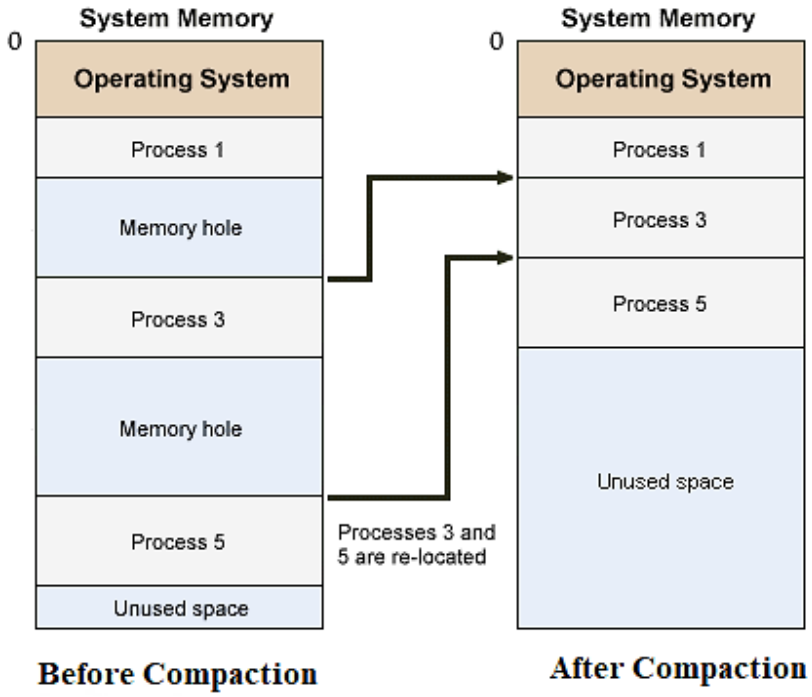


Fig.4.10: मेमरी कॉम्पॅक्शन (Memory Compaction)

कॉम्पॅक्शनचे फायदे (Advantages of Compaction)

1. एक्सटर्नल फ्रॅगमेंटेशन कमी करते.
2. मेमरी वापर कार्यक्षम बनवते.
3. मेमरी कॉन्टिग्युअस होते.
4. मेमरी कॉन्टिग्युअस झाल्यामुळे मेमरीमध्ये अधिक प्रोसेसेस लोड केल्या जाऊ शकतात, ज्यामुळे ओएसची स्केलेबिलिटी वाढते.
5. कॉम्पॅक्शनद्वारे फाइल सिस्टमचे फ्रॅगमेंटेशन तात्पुरते काढून टाकता येते.
6. मेमरी ब्लॉक्समधील अंतर कमी असल्याने मेमरी वापर सुधारतो.

कॉम्पॅक्शनचे तोटे (Disadvantages of Compaction)

1. सिस्टम कार्यक्षमता कमी होते आणि विलंब वाढतो.
2. कॉम्पॅक्शन करण्यात बराच वेळ वाया जातो.
3. सीपीयू बराच काळ निष्क्रिय बसतो.
4. कॉम्पॅक्शन करणे नेहमीच सोपे नसते.
5. मेमरी वाटप प्रक्रियेत अडथळा निर्माण झाल्यामुळे ते डेडलॉक होऊ शकते.

4.6 फ्रॅगमेंटेशन (Fragmentation)

मेमरीमधून प्रोसेस लोड केल्या जातात आणि नंतर काढून टाकल्या जातात, तेव्हा फ्री मेमरी स्पेस लहान तुकड्यांमध्ये शिल्लक राहते. कधीकधी असे होते की प्रोसेस मेमरी ब्लॉक्सना वाटल्या जाऊ शकत नाहीत कारण त्यांचा आकार लहान असतो आणि मेमरी ब्लॉक्स वापरात नसतात. ही समस्या फ्रॅगमेंटेशन म्हणून ओळखली जाते. फ्रॅगमेंटेशन दोन प्रकारचे असते –

4.6.1 इंटर्नल फ्रॅगमेंटेशन (Internal Fragmentation)

जेव्हा मेमरी ब्लॉक्स त्यांच्या विनंती केलेल्या आकारापेक्षा जास्त प्रमाणात प्रोसेसला वाटप केले जातात तेव्हा इंटर्नल फ्रॅगमेंटेशन होते. यामुळे काही न वापरलेली स्पेस शिल्लक राहते आणि इंटर्नल फ्रॅगमेंटेशन समस्या निर्माण होते.

उदाहरण: समजा मेमरी वाटपासाठी एक फिक्स्ड पार्टीशन वापरले जात आहे आणि मेमरीमध्ये वेगवेगळ्या आकाराचे ब्लॉक्स 200 एमबी स्पेस आहे. आता 35 एमबी आणि 95 एमबी आकाराची एक नवीन प्रोसेस येते आणि मेमरीचा ब्लॉक मागते. त्याला 200 एमबीचा मेमरी ब्लॉक मिळतो परंतु 165 एमबीचा मेमरी ब्लॉक वाया जातो आणि तो इतर प्रोसेसेसनाही वाटप करता येत नाही. यालाच इंटर्नल फ्रॅगमेंटेशन म्हणतात. Fig. 4.11 मध्ये, प्रोसेसला वाटप केलेल्या मेमरी आणि आवश्यक स्पेस किंवा मेमरीमधील फरकाला इंटर्नल फ्रॅगमेंटेशन म्हणतात.

Fig. 4.11: इंटर्नल फ्रॅगमेंटेशन (Internal Fragmentation)

4.6.2 एक्सटर्नल फ्रॅगमेंटेशन (External Fragmentation)

जेव्हा सिस्टममधील मेमरी स्पेस प्रोसेसेसची आवश्यकता सहजपणे पूर्ण करू शकते, परंतु ही उपलब्ध मेमरी स्पेस नॉन-कॉन्टिग्युअस असेल,तर ती नंतर वापरली जाऊ शकत नाही. मग या समस्येला एक्सटर्नल फ्रॅगमेंटेशन असे म्हणतात. समजा, 3 एमबी, 10 एमबी आणि 8 एमबी आकाराच्या मेमरीमधील तीन पार्टीशन आहे आणि 2 एमबी आकाराची प्रोसेस 3 एमबीच्या पहिल्या पार्टीशनमध्ये लोड केली गेली तर 1 एमबी मेमरी स्पेस वाया जाईल. त्याच प्रकारे, 4 एमबी आकाराची दुसरी प्रोसेस 10 बीएमच्या दुसऱ्या पार्टीशनमध्ये लोड केली, तर 6 एमबी मेमरी स्पेस वाया जाईल. पुन्हा 6 एमबी आकाराची तिसरी प्रोसेस तिसऱ्या पार्टीशन मध्ये लोड केली तर 2 एमबी मेमरी स्पेस वाया जाईल. Fig. 4.12 मध्ये दाखवल्याप्रमाणे या वाया गेलेल्या मेमरी स्पेसला एक्सटर्नल फ्रॅगमेंटेशन म्हणतात जे कॉन्टिग्युअस नसतात आणि दुसऱ्या प्रोसेससाठी वापरता येत नाहीत. एक्सटर्नल फ्रॅगमेंटेशनच्या समस्येवर कॉम्पॅक्शन हा एक उपाय आहे परंतु कॉम्पॅक्शन नेहमीच शक्य नसते. समजा जर रिलोकेशन स्टॅटिक असेल आणि लोड करते वेळी केले असेल तर त्या बाबतीत कॉम्पॅक्शन करता येणार नाही. कारण कॉम्पॅक्शन फक्त तेव्हाच शक्य आहे जेव्हा रिलोकेशन डायनॅमिक सेल आणि एक्झिक्युशनच्या वेळी केले जाते.

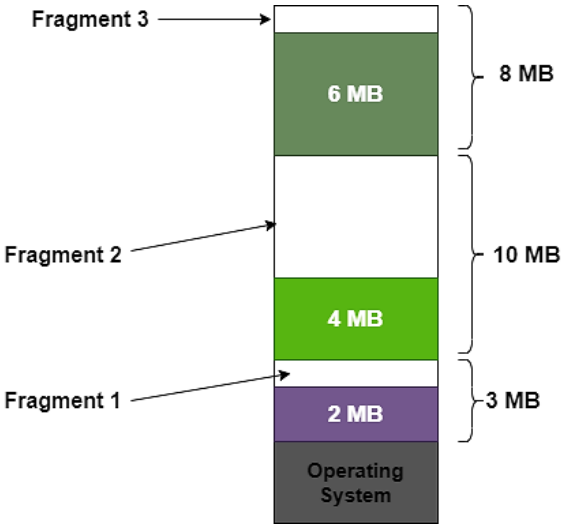


Fig. 4.12: एक्सटर्नल फ्रॅगमेंटेशन (External Fragmentation)

4.7 नॉन-कॉन्टीग्युअस मेमरी मॅनेजमेंट टेक्निक्स (Non Contiguous Memory Management Techniques)

नॉन-कॉन्टीग्युअस मेमरी मॅनेजमेंट टेक्निक्स, ज्यांना डायनॅमिक मेमरी मॅनेजमेंट टेक्निक्स असेही म्हणतात, प्रोसेसेसना नॉन-कॉन्टीग्युअस पद्धतीने मेमरी वाटप करण्याची परवानगी देतात. फिक्स्ड आणि व्हेरिएबल पार्टीशन सारख्या टेक्निक्सच्या तुलनेत मेमरी वापराच्या बाबतीत ह्या टेक्निक्स अधिक लवचिक आणि कार्यक्षम आहेत.येथे दोन सामान्य नॉन-कॉन्टीग्युअस मेमरी मॅनेजमेंट टेक्निक्स आहेत:

1. सेगमेंटेशन (Segmentation)
2. पेजिंग (Paging)

4.7.1 सेगमेंटेशन (Segmentation)

सेगमेंटेशन ही एक मेमरी मॅनेजमेंट टेक्निक आहे ज्यामध्ये प्रत्येक जॉब वेगवेगळ्या आकारांच्या अनेक सेगमेंटमध्ये विभागले जाते, प्रत्येक मॉड्यूलसाठी एक ज्यामध्ये Fig. 4.13 मध्ये दाखवल्याप्रमाणे संबंधित कार्ये करणारे पिसेस असतात. प्रत्येक सेगमेंट प्रत्यक्षात प्रोग्रामची एक वेगळी लॉजिकल ॲड्रेस स्पेस असते. जेव्हा एखादी प्रोसेस एक्झिक्युट करायची असते, तेव्हा त्याचे संबंधित सेगमेंटेशन नॉन-कॉन्टीग्युअस मेमरीमध्ये लोड केले जाते जरी प्रत्येक सेगमेंट उपलब्ध मेमरीच्या कॉन्टिग्युअस ब्लॉकमध्ये लोड केला असेल. सेगमेंटेशन मेमरी मॅनेजमेंट पेजिंगसारखेच काम करते परंतु येथे सेगमेंट व्हेरिएबल-आकाराचे असतात जिथे पेजिंगचे पेजेस फिक्स्ड आकाराचे असतात. ऑपरेटिंग सिस्टम प्रत्येक प्रोसेससाठी एक सेगमेंट मॅप टेबल आणि मुख्य मेमरीमधील सेगमेंट नंबर, त्यांचा आकार आणि संबंधित मेमरी लोकेशनसह फ्री मेमरी ब्लॉक्सची यादी ठेवते. प्रत्येक सेगमेंटसाठी, टेबल सेगमेंटचा सुरुवातीचा ॲड्रेस आणि सेगमेंटची लांबी संग्रहित करते. मेमरी स्थानाच्या संदर्भात एक व्हॅल्यू समाविष्ट असते जे सेगमेंट आणि ऑफसेट ओळखते.

सेगमेंट टेबल किंवा सेगमेंट मॅप टेबल (Segment Table or Segment Map Table): ते टू डायमेन्शनल लॉजिकल ॲड्रेस ला एक-डायमेन्शनल फिजिकल ॲड्रेस मध्ये मॅप करते. प्रत्येक टेबल एंट्रीमध्ये हे असते:

1. **बेस ॲड्रेस** (**Base Address)** : त्यात सुरुवातीचा फिजिकल ॲड्रेस असतो जिथे सेगमेंट मेमरीमध्ये असते.
2. **लिमिट** (**Limit)**: ते सेगमेंटची लांबी निर्दिष्ट करते.

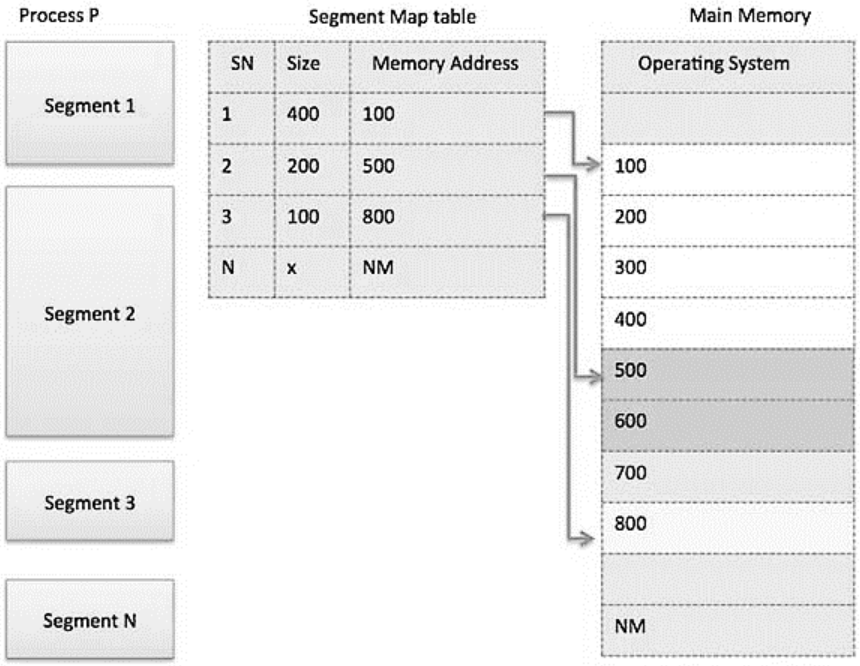


Fig. 4.13: सेगमेंटेशन (Segmentation)

टू डायमेन्शनल लॉजिकल ॲड्रेस चे एक-डायमेन्शनल फिजिकल ॲड्रेस मध्ये ट्रान्सलेशन Fig. 4.14 मध्ये दाखवले आहे. CPU द्वारे तयार केलेला ॲड्रेस यामध्ये विभागलेला आहे:

- सेगमेंट नंबर (s): सेगमेंटचे प्रतिनिधित्व करण्यासाठी आवश्यक असलेल्या बिट्सची संख्या.
- सेगमेंट ऑफसेट (d): सेगमेंटचा आकार दर्शवण्यासाठी आवश्यक असलेल्या बिट्सची संख्या.

सेगमेंट टेबलमधून लिमिट आणि बेस ॲड्रेस व्हॅल्यू मिळविण्यासाठी सेगमेंट नंबर (s) वापरला जातो. जर सेगमेंट ऑफसेट (d) सेगमेंट टेबलमधील लिमिट व्हॅल्यूपेक्षा कमी असेल तर सेगमेंट टेबलमधून परत आलेला बेस ॲड्रेस , सेगमेंटच्या सुरुवातीला पॉईंट करतो. लिमिट व्हॅल्यू फजिकल मेमरीमध्ये सेगमेंटच्या शेवट निर्देशित करते.

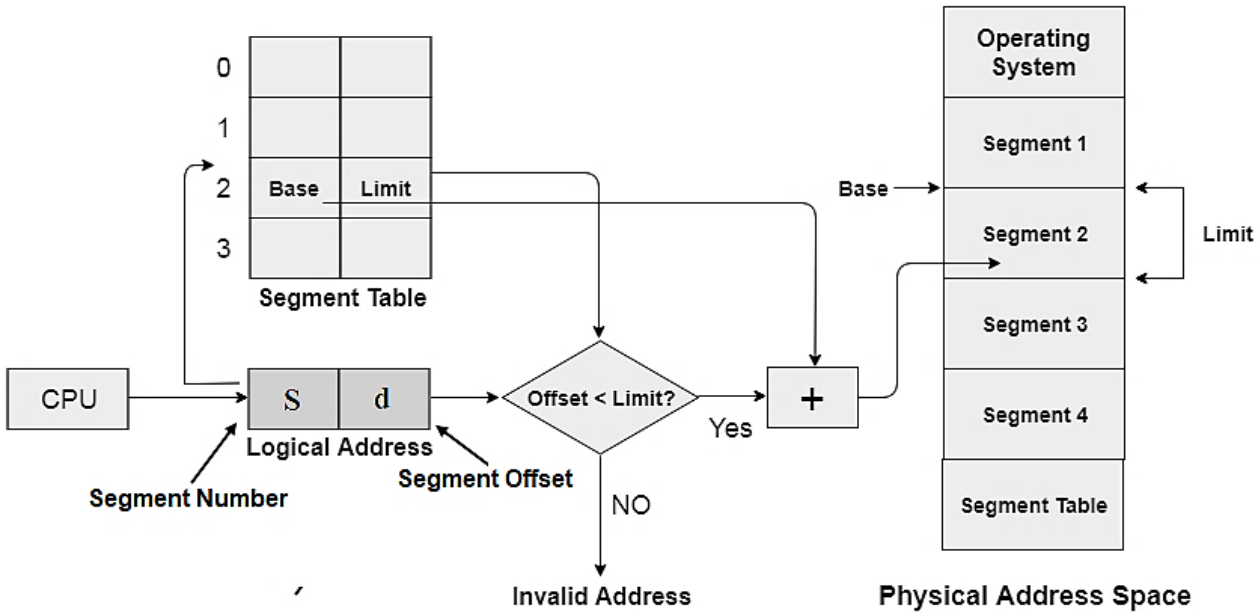


Fig. 4.14: सेगमेंटेशन मध्ये ॲड्रेस ट्रान्सलेशन (Address translation in Segmentation)

सेगमेंटेशनचे फायदे (Advantages of Segmentation)

1. इंटर्नल फ्रॅगमेंटेशन नाही आणि इंटर्नल फ्रॅगमेंटेशन कमी करते.
2. सेगमेंट टेबलचा वापर सेगमेंटच्या रेकॉर्ड साठवण्यासाठी केला जातो.
3. एकाच वेळी संपूर्ण सेगमेंट लोड केल्याने CPU कामगिरी सुधारू शकते.
4. पेजिंगमधील पेज टेबलच्या तुलनेत सेगमेंट टेबल स्वतःच कमी मेमरी वापरते.
5. सेगमेंटेशनमुळे CPU चा चांगला वापर होतो कारण संपूर्ण मॉड्यूल एकाच वेळी लोड केला जातो.
6. सेगमेंटेशन हे यूजर्सच्या भौतिक मेमरीच्या दृष्टिकोनाच्या जवळ आहे.
7. सेगमेंटेशन यूजर्सना यूजर प्रोग्राम्सना मॉड्यूलमध्ये पार्टीशन करण्याची परवानगी देते आणि हे मॉड्यूल करंट प्रोसेसचे स्वतंत्र कोड आहेत.
8. यूजर सेगमेंटचा आकार निर्दिष्ट करतो परंतु पेजिंगमध्ये, हार्डवेअर पेजचा आकार ठरवतो.
9. प्रत्येक सेगमेंटचे स्वतःचे ऍक्सेस अधिकार असू शकतात (उदा. रीड ओन्ली (Read Only), रीड-राईट (Read-Write)), ज्यामुळे मेमरी ऍक्सेसवर नियंत्रण मिळते आणि सिस्टम सुरक्षा सुधारते जी संवेदनशील डेटा किंवा कोडमध्ये अनधिकृत ऍक्सेस रोखण्यास मदत करते.
10. सेगमेंट प्रोसेससमध्ये शेअर केले जाऊ शकतात, ज्यामुळे मेमरीचा अधिक कार्यक्षम वापर शक्य होतो, विशेषतः शेअर्ड लायब्ररी किंवा कोडसाठी.

सेगमेंटेशनचे तोटे (Disadvantages of Segmentation)

1. प्रोसेससेसच्या स्वॅपिंग दरम्यान फ्री मेमरी स्पेस लहान तुकड्यांमध्ये होते, जी सेगमेंटेशन मध्ये एक मोठी समस्या आहे.
2. प्रोसेससेस लोड केल्या जातात आणि मेमरीमधून काढून टाकल्या जातात, तेव्हा फ्री मेमरी स्पेस लहान तुकड्यांमध्ये शिल्लक राहते, ज्यामुळे एक्सटर्नल फ्रॅगमेंटेशन होते.
3. इंस्ट्रक्शन्स किंवा सेगमेंट्स मेमोरीमध्ये आणण्यासाठी वेळ लागतो.
4. असमान आकाराच्या सेगमेंट्सची स्वॅपिंग करणे सोपे नाही.
5. प्रत्येक प्रोसेससेसाठी सेगमेंट टेबल राखण्याचा ओव्हरहेड देखील येतो.

4.7.2 पेजिंग (Paging)

पेजिंग ही एक मेमरी मॅनेजमेंट टेक्निक आहे जिथे ऑपरेटिंग सिस्टम प्रोसेस आणि मेन मेमरी दोन्ही अनुक्रमे पेज आणि फ्रेम्स नावाच्या फिक्स्ड -आकाराच्या युनिट्समध्ये विभाजित करते. हे प्रोसेसेसना नॉन-कॉन्टीग्युअस मेमरी लोकेशन्समध्ये लोड करण्यास अनुमती देते, ज्यामुळे एक्सटर्नल फ्रॅगमेंटेशन दूर होते. पेजिंगमध्ये, सेकंडरी मेमरी आणि मेन मेमरी समान फिक्स्ड आकाराच्या पार्टिशन्समध्ये विभागल्या जातात. सेकंडरी मेमरीच्या पार्टिशन्सना पृष्ठे म्हणतात आणि मेन मेमरीच्या पार्टिशन्सना फ्रेम्स म्हणतात. प्रत्येक प्रोसेस अशा भागांमध्ये विभागली जाते जिथे प्रत्येक भागाचा आकार पेजच्या आकाराइतकाच असतो. शेवटच्या भागाचा आकार पेजच्या आकारापेक्षा कमी असू शकतो. Fig.4.15 मध्ये दाखवल्याप्रमाणे, प्रोसेसेसची पेजेस त्यांच्या उपलब्धतेनुसार मेन मेमरीच्या फ्रेम्समध्ये साठविली जातात.

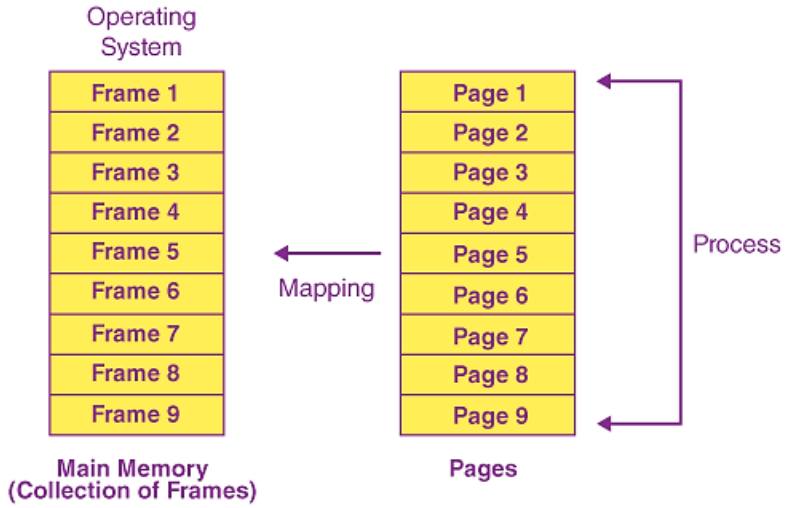


Fig. 4.15: पेजेस आणि पेज फ्रेम (Pages and Page Frame)

CPU द्वारे तयार केलेला ॲड्रेस यामध्ये विभागलेला आहे

- **पेज नंबर (Page number (p))**: लॉजिकल ॲड्रेस स्पेस किंवा पेज नंबरमधील पेजेसचे प्रतिनिधित्व करण्यासाठी आवश्यक असलेल्या बिट्सची संख्या
- **पेज ऑफसेट (Page offset (d))**: लॉजिकल ॲड्रेस स्पेस किंवा पेज ऑफसेटच्या वर्ड संख्येच्या पेज आकारात विशिष्ट वर्डचे प्रतिनिधित्व करण्यासाठी आवश्यक असलेल्या बिट्सची संख्या.

फिजिकल ॲड्रेस यामध्ये विभागलेला आहे

- **फ्रेम नंबर (Frame number (f)):** फिजिकल ॲड्रेस स्पेस किंवा फ्रेम नंबरच्या फ्रेमचे प्रतिनिधित्व करण्यासाठी आवश्यक असलेल्या बिट्सची संख्या.
- **फ्रेम ऑफसेट (Frame offset (d)):** फ्रेम किंवा फ्रेम ऑफसेटच्या फिजिकल ॲड्रेस स्पेस किंवा वर्ड संख्येच्या फ्रेम आकारात विशिष्ट वर्डचे प्रतिनिधित्व करण्यासाठी आवश्यक असलेल्या बिट्सची संख्या.

पेज टेबलचे हार्डवेअर इम्प्लिमेंटेशन डेडिकेटेड रजिस्टर वापरून करता येते. तथापि, पेज टेबल लहान असल्यासच पेज टेबलसाठी रजिस्टरचा वापर समाधानकारक असतो. जर पेज टेबलमध्ये मोठ्या संख्येने एन्ट्रीज असतील तर आपण Fig. 4.16 मध्ये दाखवल्याप्रमाणे TLB (ट्रान्सलेशन लुक-असाइड बफर) वापरू शकतो, जो एक विशेष, लहान, जलद लूकअप हार्डवेअर कॅशे (Cache) मेमरी आहे.

1. TLB ही असोशिएटिव्ह, हाय-स्पीड कॅशे (Cache) मेमरी आहे.
2. TLB मधील प्रत्येक एंट्रीमध्ये दोन भाग असतात: एक टॅग आणि एक व्हॅल्यू.
3. जेव्हा ही मेमरी वापरली जाते, तेव्हा एका आयटमची तुलना सर्व टॅगशी एकाच वेळी केली जाते. जर आयटम आढळला, तर संबंधित व्हॅल्यू परत केले जाते.

पेज टेबल (Page Table)

पेज टेबल ही मुख्य मेमरीमध्ये साठवलेला डेटा स्ट्रक्चर आहे. ते CPU द्वारे संदर्भित केलेल्या पेज नंबरला त्या फ्रेम नंबरशी मॅप करते जिथे ते पेज साठवले जाते. पेज टेबलमधील एंट्रीजची संख्या ही प्रोसेसचा विभागलेल्या पेजेसच्या संख्येइतकी असते. प्रत्येक प्रोसेसेसचे स्वतःचे स्वतंत्र पेज टेबल असते. पेज टेबल बेस रजिस्टर (PTBR) पेज टेबलचा बेस ॲड्रेस प्रदान

करते. पेज टेबलचा बेस ॲड्रेस CPU द्वारे संदर्भित केलेल्या पेज नंबरसह जोडला जातो. ते पेज टेबलची एंट्री देते ज्यामध्ये फ्रेम नंबर असतो जिथे रेफरन्स्ड पेज साठवले जाते.

पेज टेबल एंट्री फॉरमॅट (Page Table Entry format)

पेज टेबल एंट्रीमध्ये पेजबद्दल अनेक माहिती असते. पेज टेबल एंट्रीमध्ये असलेली माहिती ऑपरेटिंग सिस्टमनुसार बदलते. पेज टेबल एंट्रीमध्ये सर्वात महत्वाची माहिती म्हणजे फ्रेम नंबर, प्रेझेंट / ॲबसेंट बिट, प्रोटेक्शन बिट, रेफरन्स बिट, डर्टी बिट (सुधारित बिट).

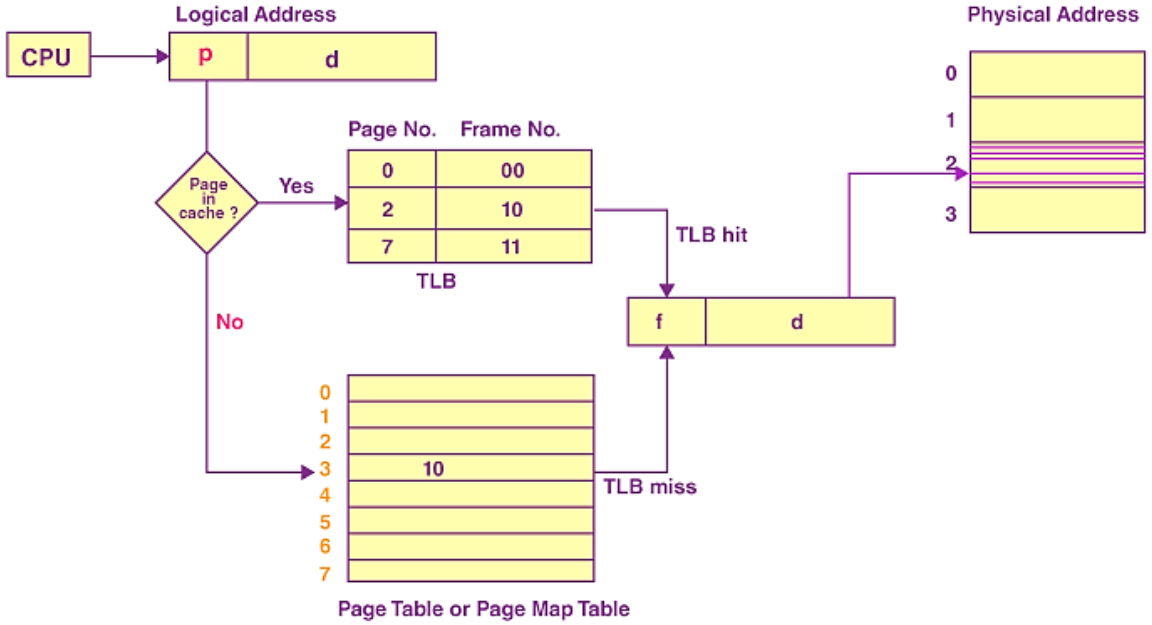


Fig. 4.17: ट्रान्सलेशन लूक असाइड बफर सहित पेजिंग (Paging with TLB)

पेजिंगचे फायदे (**Advantages of Paging**):

1. **एक्सटर्नल फ्रॅगमेंटेशन दूर करते**: पेजिंगमुळे नॉन-कॉन्टीग्युअस मेमरी अलोकेशन शक्य होते, म्हणजेच प्रोसेस सलग मेमरी ब्लॉक्समध्ये साठवण्याची आवश्यकता नसते.
2. **व्हर्च्युअल मेमरी**: उपलब्ध फिजिकल मेमरीपेक्षा मोठे प्रोग्राम रन करण्यास सक्षम करते.
3. **सुधारित मेमरी वापर**: पेजेस आणि फ्रेम फिक्स्ड आकाराचे आहेत, ज्यामुळे कार्यक्षम मेमरी अलोकेशन शक्य होते आणि वाया जाणारी जागा कमी होते.
4. **डेटा संरक्षण**: पेजिंग एकमेकांपासून प्रोसेस वेगळ्या करण्यास मदत करू शकते, एका प्रोसेसला दुसऱ्याची मेमरी खराब करण्यापासून रोखते.
5. **स्वॅपिंगची सोय**: संपूर्ण प्रोसेसवर परिणाम न करता वैयक्तिक पेजेस फिजिकल मेमरी आणि डिस्क (स्वॅप स्पेस) दरम्यान हलवता येतात, ज्यामुळे स्वॅपिंग जलद आणि अधिक कार्यक्षम होते.

पेजिंगचे तोटे (Disadvantages of Paging):

1. इंटर्नल फ्रॅगमेंटेशन: एक्सटर्नल फ्रॅगमेंटेशन टाळले जात असले तरी, जर पेज त्यामध्ये असलेल्या डेटासाठी आवश्यक असलेल्या जागेपेक्षा मोठे असेल तर काही इंटर्नल फ्रॅगमेंटेशन होऊ शकते.
2. वाढलेले ॲड्रेस ट्रान्सलेशन ओव्हरहेड: मेमरी ॲक्सेस दरम्यान पेज टेबलमध्ये ॲक्सेस करण्यासाठी वेळ लागतो, ज्यामुळे कामगिरी मंदावते.
3. पेज फॉल्ट्स: वारंवार पेज फॉल्ट्समुळे कामगिरीवर लक्षणीय परिणाम होऊ शकतो.
4. वाढलेले ओव्हरहेड: पेज टेबल राखण्यासाठी अतिरिक्त मेमरी आणि प्रोसेसिंगची आवश्यक आहे. मोठ्या प्रोसेसेससाठी, पेज टेबल लक्षणीयरीत्या वाढू शकते.
5. पेज फॉल्ट्स दरम्यान I/O ओव्हरहेड: जेव्हा आवश्यक पेज फिजिकल मेमरीमध्ये नसते (पेज फॉल्ट), तेव्हा ते सेकंडरी स्टोरेजमधून आणावे लागते, ज्यामुळे विलंब होतो आणि I/O ऑपरेशन्स वाढतात.

6. अंमलबजावणीमध्ये जटिलता: पेजिंगसाठी मेमरी मॅनेजमेंट युनिट (MMU) आणि पेज रिप्लेसमेंटसाठी अल्गोरिथमसह अत्याधुनिक हार्डवेअर आणि सॉफ्टवेअर सपोर्टची आवश्यकता असते, जे सिस्टममध्ये जटिलता वाढवते.

4.8 व्हर्च्युअल मेमरी (Virtual Memory)

व्हर्च्युअल मेमरी ही ऑपरेटिंग सिस्टीममध्ये वापरली जाणारी मेमरी मॅनेजमेंट टेक्निक आहे जी उपलब्ध असलेल्या फिजिकल रॅमपेक्षा मोठी असतानाही प्रोग्राम्स रन करण्यास अनुमती देते. रॅम आणि हार्ड डिस्क स्टोरेज दोन्ही वापरून ते फिजिकली अस्तित्वात असलेल्या मेमरी स्पेसपेक्षा मोठ्या मेमरी स्पेसचा भ्रम निर्माण करते. ऑपरेटिंग सिस्टीम व्हर्च्युअल मेमरीचे पेजेसमध्ये विभाजन करते आणि व्हर्च्युअल ॲड्रेस फिजिकल मेमरी लोकेशन्सवर मॅप करण्यासाठी पेज टेबल वापरते. जेव्हा CPU सध्या RAM मध्ये नसलेला पेज ऍक्सेस करण्याचा प्रयत्न करते, तेव्हा पेज फॉल्ट होतो. ऑपरेटिंग सिस्टीम नंतर हार्ड डिस्क (सेकंडरी स्टोरेज) मधून आवश्यक पेज मिळवते आणि ते RAM मध्ये लोड करते. व्हर्च्युअल स्टोरेजचा आकार संगणक सिस्टिमच्या ॲड्रेसिंग टेक्निकने मर्यादित असतो आणि सेकंडरी मेमरीचे प्रमाण मुख्य स्टोरेज स्थानांच्या प्रत्यक्ष संख्येने उपलब्ध नसते. हि टेक्निक हार्डवेअर आणि सॉफ्टवेअर दोन्ही वापरून इम्प्लिमेंट केली जाते. ते आकृती ४.१८ मध्ये दाखवल्याप्रमाणे, व्हर्च्युअल ॲड्रेस नावाच्या प्रोग्रामद्वारे वापरल्या जाणाऱ्या मेमरी ॲड्रेस ना संगणक मेमरीमधील फिजिकल ॲड्रेस मध्ये मॅप करते. प्रोसेसेस मधील सर्व मेमरी रेफरन्स हे लॉजिकल ॲड्रेस असतात जे रन टाइममध्ये डायनॅमिकली फिजिकल ॲड्रेस मध्ये रूपांतरित केले जातात. याचा अर्थ असा की प्रोसेस मेन मेमरीमध्ये अशा प्रकारे बदलली जाऊ शकते की ती एक्झिक्युशन दरम्यान वेगवेगळ्या वेळी मेन मेमरीमध्ये वेगवेगळ्या ठिकाणी व्यापते. एखादी प्रोसेस अनेक तुकड्यांमध्ये विभागली जाऊ शकते आणि एक्झिक्युशन दरम्यान हे तुकडे सतत मेन मेमरीमध्ये ठेवण्याची आवश्यकता नसते. जर ही वैशिष्ट्ये उपस्थित असतील, तर एक्झिक्युशन दरम्यान सर्व पेजेस किंवा सेगमेंट मेन मेमरीमध्ये असणे आवश्यक नाही. याचा अर्थ असा की आवश्यक पेजेस जेव्हा आवश्यक असतील तेव्हा मेमरीमध्ये लोड करणे आवश्यक आहे. व्हर्च्युअल मेमरी डिमांड पेजिंग (Demand Paging) किंवा डिमांड सेगमेंटेशन (Demand Segmentation)वापरून लागू केली जाते.

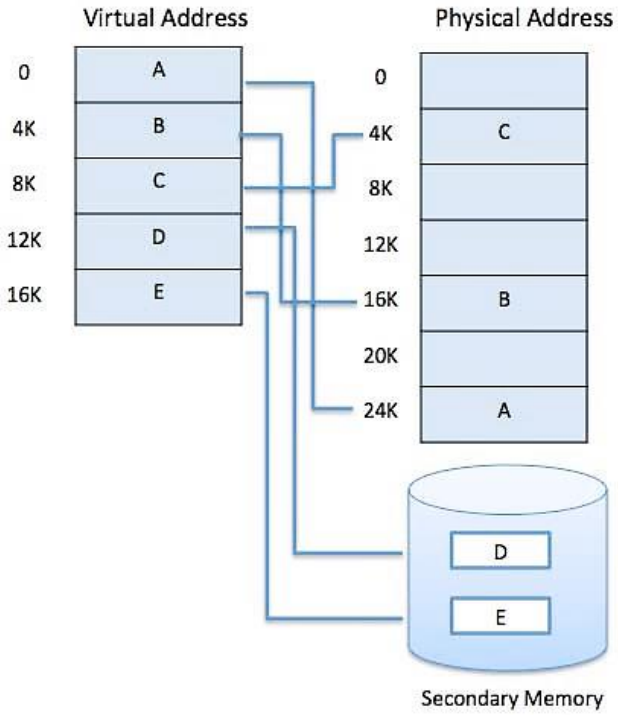


Fig.4.18: व्हर्च्युअल मेमरीचे इम्प्लिमेंटेशन (Virtual Memory Implementation)

व्हर्च्युअल मेमरीचे फायदे (Advantages of Virtual Memory)

1. मल्टीप्रोग्रामिंगची डिग्री वाढवते म्हणजेच जास्त मेमरी रॅम न खरेदी करता एकाच वेळी अनेक ऍप्लिकेशन्स रन करता येतात.

2. डेटा कॉमन असल्यास मेमरीजमध्ये डेटा शेअरिंग करता येते.
3. रिलोकेशन टाळते म्हणजेच फिजिकल मेमरीमधील कोड, कोड रिलोकेशन न करता आवश्यकतेनुसार ऍक्सेस करता येतो.
4. सिस्टममध्ये फिजिकल मेमरीपेक्षा मोठे ऍप्लिकेशन्स रन करण्यासाठी मेमरी वाढवते
5. सीपीयूचा प्रभावी वापर वाढवते

व्हर्च्युअल मेमरीचे तोटे (Disadvantages of Virtual Memory)

1. सिस्टमला हळू करते म्हणजेच स्वॅपिंगला वेळ लागतो ज्यामुळे सिस्टमची गती प्रभावित होते आणि ती मंद होते.
2. कमी हार्ड डिस्क स्पेस म्हणजेच व्हर्च्युअल मेमरी स्टोरेज स्पेस घेते जी अन्यथा दीर्घकालीन डेटा स्टोरेजसाठी वापरली जाऊ शकते.
3. सिस्टम स्थिरता कमी करते

4.8.1 डिमांड पेजिंग (Demand Paging)

डिमांड पेजिंग ही व्हर्च्युअल मेमरी सिस्टीममध्ये वापरली जाणारी एक टेक्निक आहे जिथे पेजेस फक्त CPU द्वारे विनंती केल्यावर किंवा आवश्यक असल्यासच मेन मेमरीमध्ये लोड होतात. डिमांड पेजिंगमध्ये, ऑपरेटिंग सिस्टम सुरुवातीला संपूर्ण प्रोग्राम मेमरीमध्ये लोड करण्याऐवजी रनटाइमच्या वेळी प्रोग्रामची फक्त आवश्यक पेजेस मेमरीमध्ये लोड करते. जेव्हा प्रोग्रामला सध्या मेमरीमध्ये नसलेले पेज ऍक्सेस करण्याची आवश्यकता असते तेव्हा पेज फॉल्ट होतो. ऑपरेटिंग सिस्टम नंतर डिस्कमधून मेमरीमध्ये आवश्यक पेजेस लोड करते आणि त्यानुसार पेज टेबल अपडेट करते. ही प्रोसेस रनिंग प्रोग्रामसाठी पारदर्शक असते आणि ती अशा प्रकारे रन होत राहते जणू काही पेज नेहमीच मेमरीमध्ये होते.

डिमांड पेजिंगचे फायदे (Advantages of Demand Paging)

1. फिजिकल मेमरीचा कार्यक्षम वापर
2. मोठे प्रोग्राम रन करू शकतो
3. प्रोग्राम जलद सुरू होतो.
4. मेमरी वापर कमी करते

डिमांड पेजिंगचे तोटे (Disadvantages of Demand Paging)

1. पेज फॉल्ट ओव्हरलोड
2. कामगिरी कमी होणे
3. फ्रॅगमेंटेशन होऊ शकते
4. जटिल अल्गोरिथम आवश्यक असते

4.8.2 पेज फॉल्ट (Page Fault)

"पेज मिस (Page miss)" किंवा "पेज फॉल्ट (Page fault)" हा शब्द अशा परिस्थितीला सूचित करतो जिथे संदर्भित पेज मुख्य मेमरीमध्ये आढळत नाही. जेव्हा एखादा प्रोग्राम सध्या फिजिकल मेमरी (RAM) मध्ये लोड न झालेल्या पेजवर किंवा फिक्स्ड आकाराच्या मेमरीच्या ब्लॉक ऍक्सेस करण्याचा प्रयत्न करतो, तेव्हा पेज दोष म्हणून ओळखला जाणारा अपवाद घडतो. प्रोग्रामला आवश्यक असलेल्या पेज ऍक्सेस करण्यास सक्षम करण्यापूर्वी, ऑपरेटिंग सिस्टमने पेज फॉल्ट हाताळण्यासाठी ते सेकंडरी स्टोरेज (जसे की हार्ड ड्राइव्ह) मधून मेमरीमध्ये आणले पाहिजे.

4.9 पेज रिप्लेसमेंटअल्गोरिथम (Page Replacement Algorithm)

पेज रिप्लेसमेंट अल्गोरिथम हे ऑपरेटिंग सिस्टीममध्ये वापरले जाणारे टेक्निक आहेत जे फिजिकल मेमरी भरलेली असताना मेमरी कार्यक्षमतेने मॅनेज करतात. जेव्हा एखादे नवीन पेज फिजिकल मेमरीमध्ये लोड करायचे असते आणि तिथे मोकळी जागा नसते, तेव्हा हे अल्गोरिथम कोणते विद्यमान (Current) पेज बदलायचे हे ठरवतात. जर कोणतेही पेज फ्रेम मोकळे नसेल, तर व्हर्च्युअल मेमरी मॅनेजर मेमरीमध्ये असलेल्या पेजपैकी एकाला त्या पेजने बदलण्यासाठी पेज रिप्लेसमेंट ऑपरेशन करतो ज्याच्या रेफरन्समुळे पेज फॉल्ट झाला होता. कॉमन पेज रिप्लेसमेंट टेक्निक्स आहेत जसे कि

1. फर्स्ट इन फर्स्ट आउट (First In First Out-FIFO)
2. ऑप्टिमल पेज रिप्लेसमेंट (Optimal Page replacement)
3. लिस्ट रीसेंटली यूज्ड (Least Recently Used - LRU)

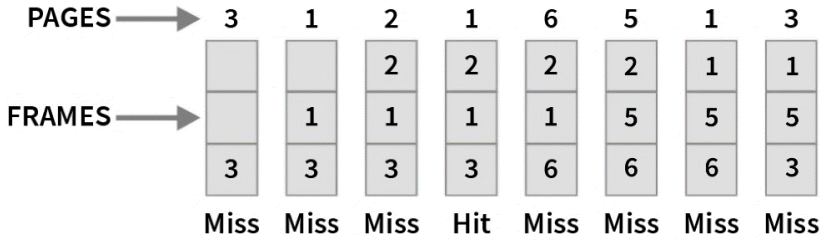
4.9.1 फर्स्ट इन फर्स्ट आउट (First In First Out - FIFO)

हे सर्वात सोपा पेज रिप्लेसमेंट अल्गोरिथम आहे. या अल्गोरिथममध्ये, ऑपरेटिंग सिस्टम क्यूतील मेमरीमधील सर्व पेजेसचा ट्रॅक ठेवते, सर्वात जुने पेज क्यूच्या समोर असते.

जेव्हा एखादे पेज बदलण्याची आवश्यकता असते तेव्हा क्यूच्या समोरील पेज काढून टाकण्यासाठी निवडले जाते.

उदाहरण 1: पेज रेफरन्स स्ट्रिंग 3, 1, 2, 1, 6, 5, 1, 3 अशी विचारात घ्या ज्यामध्ये 3 पेज फ्रेम आहेत. पेज फॉल्टची संख्या शोधा. (Consider the page reference string as 3, 1, 2, 1, 6, 5, 1, 3 with 3-page frames. Find the number of page faults)

उत्तर:



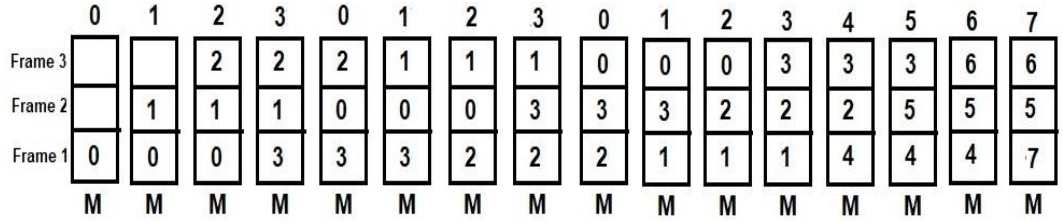
एकूण पेज फॉल्ट =7

- सुरुवातीला, सर्व स्लॉट रिकामे असतात म्हणून पेज फॉल्ट्स 3,1,2 वर येतात, म्हणून पेज फॉल्ट्स = 3
- जेव्हा पेज 1 येतो तेव्हा तो मेमरीमध्ये आहे म्हणून पेज फॉल्ट्स होत नाहीत. म्हणून पेज फॉल्ट्स = 3
- जेव्हा पेज 6 येतो तेव्हा तो मेमरीमध्ये नाही आणि पेज फॉल्ट्स होतो . कोणतेही रिक्त स्लॉट नसल्यामुळे, आपण पेज 3 काढून टाकतो आणि पेज 6 टाकतो. म्हणून, पेज फॉल्ट्स = 4
- जेव्हा पेज 5 येतो तेव्हा तो देखील मेमरीमध्ये नाही आणि म्हणून पेज फॉल्ट्स होतो . पेज १ काढून टाकले जाते आणि पेज 5 टाकले जाते. म्हणून, पेज फॉल्ट्स = 5
- जेव्हा पेज 1 येतो तेव्हा तो मेमरीमध्ये नाही आणि पुन्हा पेज फॉल्ट्स होतो. पेज 2 काढून टाकले जाते आणि पेज 1 टाकले जाते. म्हणून, पेज फॉल्ट्स = 6
- जेव्हा पेज 3 येतो तेव्हा तो मेमरीमध्ये नाही आणि पुन्हा पेज फॉल्ट्स होतो. पेज 6 काढून टाकले जाते आणि पेज 3 टाकले जाते. म्हणून, पेज फॉल्ट्स = 7

उदाहरण 2: फ्रेम आकार 3 आणि 4 सह 0 1 2 3 0 1 2 3 0 1 2 3 4 5 6 7 1 ही स्ट्रिंग विचारात घ्या, दोन्ही प्रकरणांमध्ये FIFO अल्गोरिथम वापरून पेज फॉल्टची गणना करा. (Consider the string 0 1 2 3 0 1 2 3 0 1 2 3 4 5 6 7 1, with frame size 3 and 4, calculate page fault in both the cases using FIFO algorithm.)

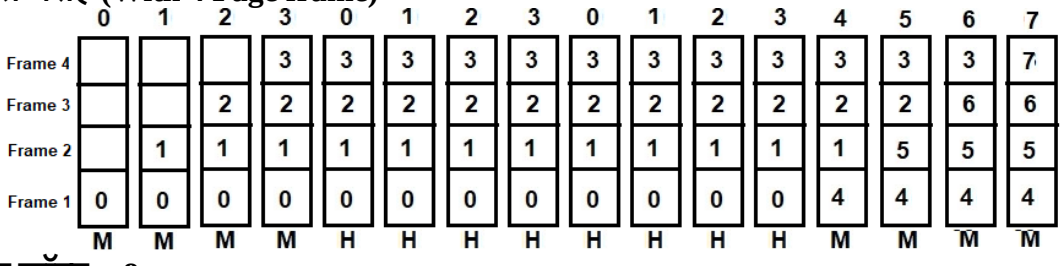
उत्तर:

a) 3 पेज फ्रेमसह (With 3 Page frame)



एकूण पेज फॉल्ट = 16

b) 4 पेज फ्रेमसह (With 4 Page frame)



एकूण पेज फॉल्ट = 8

फायदे (Advantages):

1. हे समजण्यास आणि इम्प्लिमेंट करायला सोपे आहे.
2. जास्त ओव्हरहेड होत नाही

तोटे (Disadvantages):

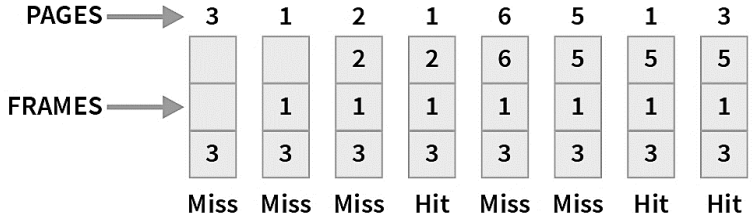
1. ते फार प्रभावी नाही म्हणून खराब कामगिरी देते
2. सिस्टमला प्रत्येक फ्रेमचा ट्रॅक ठेवणे आवश्यक आहे
3. जेव्हा आपण FIFO वापरताना फ्रेमची संख्या वाढवतो तेव्हा आपण प्रोसेसेसना अधिक मेमरी देत असतो. म्हणून, पेज फॉल्ट कमी झाला पाहिजे, परंतु येथे पेज फॉल्ट वाढतो. या समस्येला बेलाडीज अनोमली (Belady's Anomaly) म्हणतात.
4. चुकीच्या रिप्लेसमेंटच्या निवडीमुळे पेज फॉल्टचा दर वाढतो आणि प्रोसेस एक्झिक्युशन मंदावते.

4.9.2 ऑप्टिमल पेज रिप्लेसमेंट (Optimal Page replacement)

ऑप्टिमल पेज रिप्लेसमेंट अल्गोरिथम म्हणजे सर्वोत्तम पेज रिप्लेसमेंट अल्गोरिथम आहे कारण या अल्गोरिथममध्ये कमीत कमी पेज फॉल्ट्स येतात. या अल्गोरिथममध्ये, भविष्यात सर्वात जास्त काळ वापरल्या जाणार नाहीत अशा पेजने पेजेस बदलली जातात. सोप्या भाषेत, भविष्यात सर्वात जास्त काळ न वापरल्या जाणाऱ्या पेजना या अल्गोरिथममध्ये बदलल्या जातात.

उदाहरण 1: पेज रेफरन्स स्ट्रिंग 3, 1, 2, 1, 6, 5, 1, 3 अशी विचारात घ्या ज्यामध्ये 3 पेज फ्रेम आहेत. पेज फॉल्टची संख्या शोधा. (Consider page reference string 3, 1, 2, 1, 6, 5, 1, 3, calculate page fault with 3-page frames.)

उत्तर:

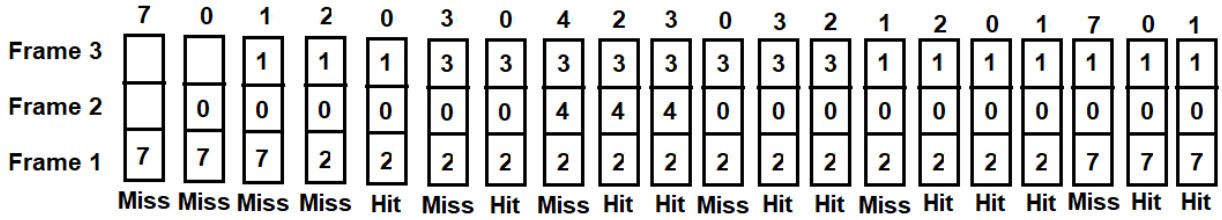


एकूण पेज फॉल्ट = 5

- सुरुवातीला, सर्व स्लॉट रिकामे असतात म्हणून पेज फॉल्ट्स 3,1,2 वर येतात, म्हणून पेज फॉल्ट्स = 3
- जेव्हा पेज 1 येतो तेव्हा तो मेमरीमध्ये आहे म्हणून पेज फॉल्ट्स होत नाहीत. म्हणून पेज फॉल्ट्स = 3
- जेव्हा पेज 6 येतो तेव्हा तो मेमरीमध्ये नाही आणि पेज फॉल्ट्स होतो . कोणतेही रिक्त स्लॉट नसल्यामुळे, आपण पेज 2 काढून टाकतो आणि पेज 6 टाकतो. म्हणून, पेज फॉल्ट्स = 4
- जेव्हा पेज 5 येतो तेव्हा तो देखील मेमरीमध्ये नाही आणि म्हणून पेज फॉल्ट्स होतो . पेज 6 काढून टाकले जाते कारण तो पुन्हा लागणार नसतो आणि पेज 5 टाकले जाते. म्हणून, पेज फॉल्ट्स = 5
- जेव्हा पेज 1 आणि 3 येतो तेव्हा ते मेमरीमध्ये आहे आणि पेज फॉल्ट्स होत नाही. म्हणून, पेज फॉल्ट्स = 5

उदाहरण 2: जर पेजेस 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 या क्रमाने येत असेल तर 3 पृष्ठ फ्रेम गृहीत धरून, ऑप्टिमल पेज रिप्लेसमेंट अल्गोरिथमचा वापर करून एकूण पेज फॉल्टची संख्या शोधा.(Find out the total number of page faults using optimal page replacement assuming 3-page frame. if the page are coming in the order 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1)

उत्तर: एकूण पेज फॉल्ट = 9

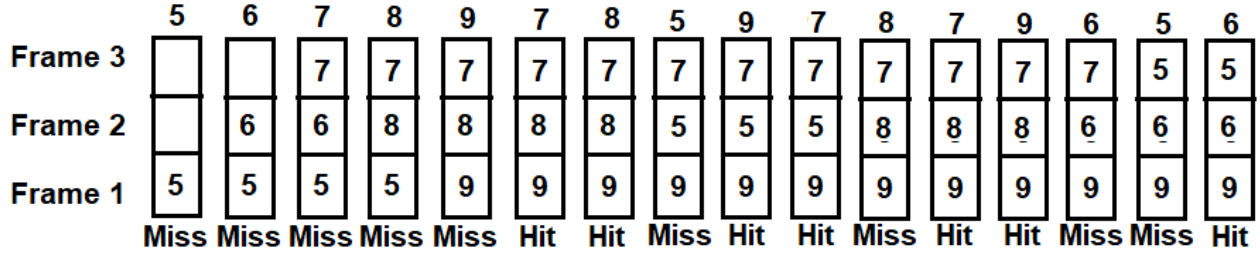


उदाहरण 3: तीन पेज फ्रेमसह खालील पेज रेफरन्स स्ट्रिंग आगमन विचारात घ्या

5 6 7 8 9 7 8 5 9 7 8 7 9 6 5 6.

ऑप्टिमल पेज रिप्लेसमेंट अल्गोरिथमचा वापर करून एकूण पेज फॉल्टची संख्या शोधा. (Consider the following page reference string arrival with three page frames 5 6 7 8 9 7 8 5 9 7 8 7 9 6 5 6 Calculate number of page faults with optimal page replacement algorithms.)

उत्तर: एकूण पेज फॉल्ट = 9



फायदे (Advantages)

1. कॉम्प्लेक्सिटी कमी आहे.
2. उत्कृष्ट कार्यक्षमता
3. साध्या डेटा स्ट्रक्चर्सचा वापर इम्प्लिमेंट करण्यासाठी साठी केला जाऊ शकतो
4. सर्वात कमी पेज फॉल्ट रेट.
5. बेलाडीज अनोमली (Belady's Anomaly) परिणामाचा कधीही सामना करावा लागत नाही.

तोटे (Disadvantages)

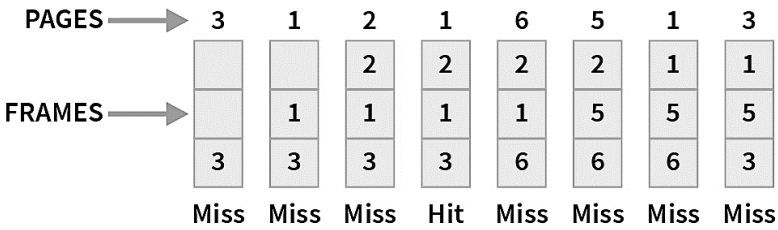
1. भविष्यात प्रोग्राम्सची जाणीव असणे आवश्यक आहे, जे प्रत्येक वेळी शक्य नाही
2. जास्त वेळ घेणारे
3. त्रुटी हाताळणे कठीण आहे.

4.9.3 लिस्ट रीसेंटली यूज्ड (Least Recently Used - LRU)

लिस्ट रीसेंटली यूज्ड पेज रिप्लेसमेंट अल्गोरिथम मागचा कालावधीत पेजेसच्या वापराचा ट्रॅक ठेवते. हे अल्गोरिथम रेफ्रन्सची लोकॅलिटीच्या तत्त्वावर आधारित आहे जे सांगते की प्रोग्राममध्ये कमी कालावधीत त्याच मेमरी लोकेशन्सच्या सेटवर पुनरावृत्ती करण्याची टेंडन्सी असते. म्हणून भूतकाळात जास्त वापरल्या गेलेल्या पेजेसचा भविष्यातही जास्त वापर होण्याची शक्यता जास्त असते. या अल्गोरिथममध्ये, जेव्हा पेज फॉल्ट होतो, तेव्हा नवीन विनंती केलेले पेज त्या पेजची जागा घेते जे जास्त काळ वापरले गेले नाही.

उदाहरण 1: पेज रेफरन्स स्ट्रिंग 3, 1, 2, 1, 6, 5, 1, 3 अशी विचारात घ्या ज्यामध्ये 3 पेज फ्रेम आहेत. पेज फॉल्टची संख्या शोधा. (Consider page reference string 3, 1, 2, 1, 6, 5, 1, 3, calculate page fault with 3-page frames.)

उत्तर:



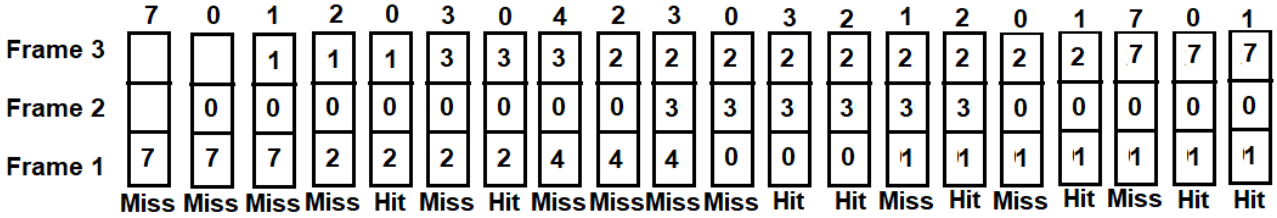
- सुरुवातीला, सर्व स्लॉट रिकामे असतात म्हणून पेज फॉल्ट्स 3,1,2 वर येतात, म्हणून पेज फॉल्ट्स = 3
- जेव्हा पेज 1 येतो तेव्हा तो मेमरीमध्ये आहे म्हणून पेज फॉल्ट्स होत नाहीत. म्हणून पेज फॉल्ट्स = 3
- जेव्हा पेज 6 येतो तेव्हा तो मेमरीमध्ये नाही आणि पेज फॉल्ट्स होतो . कोणतेही रिक्त स्लॉट नसल्यामुळे, आपण लिस्ट रीसेंटली यूज्ड पेज 3 काढून टाकतो आणि पेज 6 टाकतो. म्हणून, पेज फॉल्ट्स = 4

- जेव्हा पेज 5 येतो तेव्हा तो देखील मेमरीमध्ये नाही आणि म्हणून पेज फॉल्ट्स होतो . पेज 1 काढून टाकले जाते कारण लिस्ट रीसेंटली यूज्ड पेज आहे आणि पेज 5 टाकले जाते. म्हणून, पेज फॉल्ट्स = ५
- जेव्हा पेज 1 येतो तेव्हा तो देखील मेमरीमध्ये नाही आणि म्हणून पेज फॉल्ट्स होतो . पेज 2 काढून टाकले जाते कारण लिस्ट रीसेंटली यूज्ड पेज आहे आणि पेज 5 टाकले जाते. म्हणून, पेज फॉल्ट्स = ६
- जेव्हा पेज 3 येतो तेव्हा तो देखील मेमरीमध्ये नाही आणि म्हणून पेज फॉल्ट्स होतो . पेज 6 काढून टाकले जाते कारण लिस्ट रीसेंटली यूज्ड पेज आहे आणि पेज 3 टाकले जाते. म्हणून, पेज फॉल्ट्स = 7

एकूण पेज फॉल्ट = 7

उदाहरण 2: जर पेजेस 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 या क्रमाने येत असेल तर 3 पृष्ठ फ्रेम गृहीत धरून, लिस्ट रीसेंटली यूज्ड अल्गोरिथमचा वापर करून एकूण पेज फॉल्टची संख्या शोधा.(Find out the total number of page faults using optimal page replacement assuming 3-page frame. if the page are coming in the order 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1)

उत्तर: एकूण पेज फॉल्ट = 12



फायदे (Advantages)

1. ते पूर्ण विश्लेषणासाठी खुले आहे.
2. यामध्ये, आपण अलीकडे लिस्ट रीसेंटली यूज्ड पेजची जागा घेतो, त्यामुळे बेलाडीज ॲनोमलीपासून (Belady's Anomaly) मुक्त होतो.
3. निवडण्यास सोपे असलेले पेज जे फॉल्ट झाले आहे आणि बऱ्याच काळापासून वापरले गेले नाही म्हणून अधिक कार्यक्षम आहे

तोटे (Disadvantages)

1. यासाठी अतिरिक्त डेटा स्ट्रक्चर अंमलात आणणे आवश्यक आहे.
2. हार्डवेअर सहाय्य अत्यंत आवश्यक आहे.
3. अधिक जटिल

References:

1. Operating System: A Concept-Based Approach by Dhananjay M. Dhamdhere, McGraw Hill Education 3rd edition, ISBN: 978-1259005589
2. Operating Systems : Internals and Design Principles by William Stallings, Pearson Education 9th Edition, ISBN: 978-9352866717
3. Linux The Complete Reference by Richard Petersen, McGraw Hill, 6th edition, ISBN: 978-0071492478
4. Linux command line and shell scripting by Richard Blum, Wiley India, ISBN: 978-1118983843
5. Operating System Concepts by Abraham Silberschatz and James Peterson, Wiley India, ISBN: 9781119454083

Websites:

1. https://www.geeksforgeeks.org/memory-management-in-operating-system/
2. https://www.javatpoint.com/memory-management-operating-system
3. https://www.tutorialspoint.com/operating_system/os_memory_management.htm

4. https://www.geeksforgeeks.org/virtual-memory-in-operating-system/
5. https://www.geeksforgeeks.org/paging-in-operating-system/
6. https://www.geeksforgeeks.org/segmentation-in-operating-system/
7. https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/
8. https://www.javatpoint.com/os-free-space-management
9. https://www.geeksforgeeks.org/free-space-management-in-operating-system/

E-learning material (Video lectures)

1. https://www.youtube.com/watch?v=FrTttJLN7Kw – Memory Management
2. https://www.youtube.com/watch?v=W0068fRJTGQ - Memory Management
3. https://www.youtube.com/watch?v=FrTttJLN7Kw - Memory Management
4. https://www.youtube.com/watch?v=o2_iCzS9-ZQ – Virtual Memory
5. https://www.youtube.com/watch?v=NPhcwfnYZd8 - Virtual Memory
6. https://www.youtube.com/watch?v=kCmu5k6jfXM - Virtual Memory
7. https://www.youtube.com/watch?v=6c-mOFZwP_8 – Paging
8. https://www.youtube.com/watch?v=2pgI46Q72fA – Paging
9. https://www.youtube.com/watch?v=dz9Tk6KCMlQ -Segmentation
10. https://www.youtube.com/watch?v=Ge8Ix11rVMs – Segmentation
11. https://www.youtube.com/watch?v=ALahNOwrTvg - Fragmentation
12. https://www.youtube.com/watch?v=bK-VhQA512c - Fragmentation
13. https://www.youtube.com/watch?v=8rcUs5RutX0 – Page replacement Algorithm
14. https://www.youtube.com/watch?v=16kaPQtYo28 – Page replacement Algorithm

युनिट-5
फाइल मॅनेजमेंट
(File Management)

विषय निष्पत्ती (Course Outcome):

CO5: ऑपरेटिंग सिस्टममध्ये प्रभावी फाइल मॅनेजमेंटसाठी टेक्निक्स लागू करा.

घटक निष्पत्ती (Theory Learning Outcome-TLO):

1. दिलेल्या फाइल सिस्टमची रचना उदाहरणासह स्पष्ट करा.
2. फाइल ऍक्सेस पद्धतीचे मेकॅनिझम स्पष्ट करा.
3. ऍक्सेस डायरेक्टरीज तयार करण्याची आणि दिलेल्या फाइल ऍक्सेस परवानग्या देण्याची प्रक्रिया स्पष्ट करा.

5.1 फाईल्स (Files)

संकल्पना (Concepts)

फाइल हा संबंधित इन्फॉर्मेशनचा नाव दिलेला संग्रह आहे जो सेकंडरी स्टोरेजवर रेकॉर्ड केला जातो जसे की मॅग्नेटिक डिस्क, मॅग्नेटिक टेप आणि ऑप्टिकल डिस्क. सर्वसाधारणपणे, फाइल म्हणजे बिट्स, बाइट्स, लाईन्स किंवा रेकॉर्डचा एक क्रम ज्याचा अर्थ फाइल निर्माता आणि यूजरद्वारे परिभाषित केला जातो.

5.1.1 फाईलचे ऍट्रिब्युटस (Attributes of a File)

फाईलचे काही ऍट्रिब्युटस खालीलप्रमाणे आहेत:

- **नाव (Name)**: ही एकमेव माहिती आहे जी मानवी-वाचनीय स्वरूपात आहे.
- **आयडेंटिफायर (Identifier)**: फाइल फाइल सिस्टममध्ये एक युनिक टॅग (क्रमांक) द्वारे ओळखली जाते.
- **प्रकार (Type)**: वेगवेगळ्या प्रकारच्या फायलींना समर्थन देणाऱ्या सिस्टमसाठी हे आवश्यक आहे.
- **स्थान (Location)**: डिव्हाइसवर स्थान फाइल पॉईंट करण्यासाठी पॉईंटर.
- **आकार (Size)** फाईलचा सध्याचा आकार.
- **संरक्षण (Protection)**: हे वाचन, लेखन, कार्यवाही करण्याची शक्ती नियंत्रित करते आणि नियुक्त करते.
- **वेळ**, **तारीख आणि यूजर इडेंटिफिकेशन (Time, date, and user identification)**: संरक्षण, सुरक्षा आणि वापर देखरेखीसाठी हा डेटा आहे.

5.1.2 फाइल ऑपरेशन्स (File Operations)

ऑपरेटिंग सिस्टममध्ये खाली दिलेली मूलभूत फाइल ऑपरेशन्स करता येते-

1. **फाइल तयार करणे (Creating a file)**: फाइल तयार करण्यासाठी दोन स्टेप्स आवश्यक आहेत. प्रथम, फाइल सिस्टममध्ये फाइलसाठी जागा शोधावी लागते. दुसरे म्हणजे, नवीन फाइलसाठी डायरेक्टरीमध्ये एक एंट्री करावी लागते.
2. **फाइलमध्ये लिहिणे (Writing a file)**: फाइलमध्ये लिहिण्यासाठी, आपण सिस्टम कॉल करतो ज्यामध्ये फाइलचे नाव आणि फाइलमध्ये लिहायची माहिती दोन्ही निर्दिष्ट केले जातात. फाइलचे नाव दिल्यास, सिस्टम फाइलचे स्थान शोधण्यासाठी डायरेक्टरी शोधते. सिस्टमने फाइलमधील त्या स्थानावर एक राइट पॉइंटर (Write Pointer) ठेवला पाहिजे जिथे पुढील राइट होणार आहे. जेव्हा जेव्हा राइट (Write) होईल तेव्हा राइट पॉइंटर (Write Pointer) अपडेट केला पाहिजे.
3. **फाइलमधून वाचणे (Reading a file)**: फाइलमधून वाचण्यासाठी, आपण सिस्टम कॉल वापरतो जो फाइलचे नाव आणि फाइलचा पुढील ब्लॉक (मेमरीमध्ये) कुठे ठेवायचा हे निर्दिष्ट करतो. पुन्हा, डायरेक्टरी संबंधित एंट्रीसाठी शोधली जाते आणि सिस्टमला फाइलमधील त्या स्थानावर एक रीड पॉइंटर (Read Pointer) ठेवला पाहिजे जिथे पुढील रीड (Read) होणार आहे. एकदा वाचन झाले की, रीड पॉइंटर अपडेट केला जातो.
4. **फाइलमध्ये रिपोझिशनिंग करणे (Repositioning within a file)**: योग्य एंट्रीसाठी डायरेक्टरी शोधली जाते आणि करंट-फाइल-पोझिशन पॉइंटर दिलेल्या व्हॅल्यूवर रिपोझिशन केला जातो. फाइलमध्ये रिपोझिशन करण्यासाठी कोणत्याही वास्तविक I/O चा वापर करण्याची आवश्यकता नाही. या फाइल ऑपरेशनला फाइल शोध असेही म्हणतात.

5. **फाइल डिलीट करणे (Deleting a file)**: फाइल डिलीट करण्यासाठी, नाव दिलेल्या फाइलसाठी डायरेक्टरी शोधतो. संबंधित डायरेक्टरी एंट्री सापडल्यानंतर, सर्व फाइल स्पेस रिकामी करतो, जेणेकरून ती इतर फाइल्ससाठी पुन्हा वापरली जाऊ शकते आणि डायरेक्टरी एंट्री मिटवतो..
6. **संरक्षण (Protection):** ऍक्सेस-कंट्रोल माहिती हे ठरवते की वाचन (Reading), लेखन (Writing), एक्झिक्युट करणे कोण करू शकते.
7. **फाईल ट्रन्केट करणे (Truncating a file):** यूजरला फाईलमधील सामग्री मिटवण्याची इच्छा असू शकते परंतु त्याचे ऍट्रिब्युटस ठेवू शकतात. युजरला फाईल हटविण्यास भाग पाडण्याऐवजी आणि नंतर ती पुन्हा तयार करण्याऐवजी, हे कार्य सर्व ऍट्रिब्युटसना अपरिवर्तित राहू देते - फाईलची लांबी वगळता - परंतु फाईलची लांबी शून्य आणि त्याच्या फाईल स्पेसमध्ये रीसेट होऊ देते.

5.1.3 फायलींचे प्रकार (Types of Files)

ऑपरेटिंग सिस्टममध्ये, फाइल्सचे वर्गीकरण सामान्यतः रेगुलर फाइल्स, विशेष फाइल्स आणि डिरेक्टरी फाइल्समध्ये केले जाते. रेगुलर फाइल्स डेटा संग्रहित करतात, जसे की टेक्स्ट फाइल्स, डॉक्युमेंट्स किंवा इमेज फाईल्स, तर विशेष फाइल्स हार्डवेअर डिव्हाइसेसचे प्रतिनिधित्व करतात आणि डिरेक्टरी फाइल्स सिस्टममधील इतर फाइल्स व्यवस्थापित करतात.

1. **रेगुलर फाइल्स** (**Regular Files)**: डेटा आणि ऍप्लिकेशन्स संग्रहित करण्यासाठी वापरल्या जाणाऱ्या या सर्वात सामान्य प्रकारच्या फाइल आहेत. त्या टेक्स्ट फाइल्स, बायनरी फाइल्स, एक्झिक्युटेबल प्रोग्राम्स किंवा ऑपरेटिंग सिस्टम मॅनेज करू शकणाऱ्या इतर कोणत्याही प्रकारच्या डेटा असू शकतात.

उदाहरणे:

1. टेक्स्ट फाइल्स (Text files): .txt, .doc, .docx, .pdf
2. इमेज फाईल्स (Image files): .jpg, .png, .gif
3. एक्झिक्युटेबल फाईल्स (Executable files): .exe, .com, .sh, .drv
4. ऍप्लिकेशन्स डेटा फाईल्स (Application data files): .dat, .db
5. सोअर्स कोड फाईल्स (Source code files): .c, .cpp, .java, .py
6. मीडिया (Media): img, mp3, mp4, jpg, png, flac, etc.

2. विशेष फायली: या फायली फिजिकल हार्डवेअर डिव्हाइसेस किंवा इंटरफेस दर्शवतात, ज्यामुळे हार्डवेअरशी परस्परसंवाद साधता येतो. त्यांना डिव्हाइस फाइल्स असेही म्हणतात.

उदाहरणे:

1. प्रिंटर डिवाइस फाईल (Printer device file): /dev/lp0
2. कीबोर्ड डिवाइस फाईल (Keyboard device file): /dev/input/by-id/usb-0900.0000-event0
3. हार्ड डिस्क डिवाइस फाईल (Hard disk device file): /dev/sda
4. टर्मिनल डिवाइस फाईल (Terminal device file): /dev/tty

3. डिरेक्टरी फाइल्स: डिरेक्टरी फाइल्सचा वापर इतर फाइल्सना एका श्रेणीबद्ध रचनेत व्यवस्थित करण्यासाठी केला जातो. त्या इतर फाइल्स आणि सबडिरेक्टरीज साठवण्यासाठी कंटेनर म्हणून काम करतात.

उदाहरणे:

UNIX मध्ये

1. /home/user/documents
2. /usr/bin
3. /var/log

विंडोजमध्ये

1. C:\WINDOWS\SYSTEM32
2. D:\TCC
3. E:\TASM

5.1.4 फाइल सिस्टम स्ट्रक्चर (File System Structure)

ऑपरेटिंग सिस्टीममध्ये फाइल संकल्पनांच्या क्षेत्रात अनेक वेगवेगळ्या फाइल सिस्टीम आहेत ज्या सामान्यतः आधुनिक ऑपरेटिंग सिस्टीममध्ये वापरल्या जातात.

1. **फ्लॅट फाइल सिस्टीम (Flat File System)**: एक साधी रचना जिथे सर्व फाइल्स एकाच पातळीवर साठवल्या जातात, कोणत्याही डिरेक्टरी हाइरार्कि शिवाय. ही एक जुनी पद्धत आहे आणि सामान्यतः वापरली जात नाही.
2. **हाइरार्किकल फाइल सिस्टीम (Hierarchical File System)**: सर्वात सामान्य रचना, जिथे फाइल्स आणि डिरेक्टरीज ट्री सारख्या स्वरूपात व्यवस्थित केल्या जातात. हे सोपे ऑर्गनायझेशन आणि नेव्हिगेशन करण्यास अनुमती देते, विंडोज सारख्या आधुनिक ऑपरेटिंग सिस्टीम NTFS आणि Linux या संरचनेचा वापर ext4 वापरून करतात.
3. **नेटवर्क फाइल सिस्टीम (Network File System-NFS):** एक प्रणाली जी नेटवर्कवरून फाइल्स ऍक्सेस करण्याची परवानगी देते, जणू काही त्या ऑपरेटिंग सिस्टीममध्ये स्थानिक आहेत.
4. **डिस्ट्रीब्युटेड फाइल सिस्टीम (Distributed File System)**: फाइल्स अनेक मशीन्समध्ये वितरित केल्या जातात, ज्यामुळे कामगिरी आणि फॉल्ट टॉलरन्स असे फायदे मिळतात.
5. **व्हर्च्युअल फाइल सिस्टीम(Virtual File System-VFS)**: एक ॲब्स्ट्रॅक्शन लेयर जो ऑपरेटिंग सिस्टीमला वेगवेगळ्या फाइल सिस्टीमसह एकसमान पद्धतीने कार्य करण्यास अनुमती देते, लवचिकता आणि स्केलेबिलिटी प्रदान करते.

फाइल सिस्टीमची उदाहरणे:

1. **FAT (फाइल अलोकेशन टेबल)**: विंडोज आणि इतर सिस्टीमच्या जुन्या आवृत्त्यांमध्ये वापरलेली जुनी फाइल सिस्टम.
2. **NTFS (न्यू टेक्नॉलॉजी फाइल सिस्टम)**: विंडोजद्वारे वापरलेली एक आधुनिक फाइल सिस्टम, जी फाइल आणि फोल्डर परवानग्या, कॉम्प्रेशन आणि एन्क्रिप्शन सारख्या वैशिष्ट्यांना समर्थन देते.
3. **ext (एक्सटेंडेड फाइल सिस्टम):** लिनक्स आणि युनिक्स-आधारित ऑपरेटिंग सिस्टमवर सामान्यतः वापरली जाणारी फाइल सिस्टम.
4. **HFS (हाइरार्किकल फाइल सिस्टम)**: मॅकओएसद्वारे वापरलेली फाइल सिस्टम.
5. **APFS (ॲपल फाइल सिस्टम):** मॅक आणि iOS डिव्हाइसेससाठी ऍपल द्वारे सादर केलेली एक नवीन फाइल सिस्टम.

फाइल स्ट्रक्चर ऑपरेटिंग सिस्टमला समजेल अशा आवश्यक फॉरमॅटनुसार असावे.

- फाइलची त्याच्या प्रकारानुसार एक विशिष्ट परिभाषित रचना असते.
- टेक्स्ट फाइल म्हणजे लाईन्स मध्ये व्यवस्थित केलेल्या कॅरेक्टर्सचा क्रम.
- सोर्स फाइल म्हणजे प्रोसिजर आणि फंक्शन्सचा क्रम.
- ऑब्जेक्ट फाइल म्हणजे मशीनला समजू शकणाऱ्या ब्लॉक्समध्ये व्यवस्थित केलेल्या बाइट्सचा क्रम.
- जेव्हा ऑपरेटिंग सिस्टम वेगवेगळ्या फाइल स्ट्रक्चर्स परिभाषित करते, तेव्हा त्यात या फाइल स्ट्रक्चर्सना सपोर्ट करण्यासाठी कोड देखील असतो. युनिक्स, एमएस-डॉस (MS-DOS) किमान फाइल स्ट्रक्चर्सना सपोर्ट करतात.

फायलींची रचना अनेक प्रकारे करता येते ज्यामध्ये तीन सामान्य स्ट्रक्चर त्यांच्या संक्षिप्त वर्णनासह खाली दिल्या आहेत.

1. **फाइल स्ट्रक्चर 1**: येथे, जसे तुम्ही Fig. 5.1 (a) वरून पाहू शकता, फाइल ही बाइट्सचा एक अन स्ट्रक्चर्ड क्रम आहे. म्हणून, OS ला फाइलमध्ये काय आहे याची पर्वा नसते , कारण ते फक्त बाइट्स पाहते.
2. **फाइल स्ट्रक्चर 2**: आता, जसे तुम्ही Fig. 5.1 (b) वरून पाहू शकता जे फाइलची दुसरी रचना दर्शवते, जिथे फाइल ही फिक्स्ड -लांबीच्या रेकॉर्डचा क्रम आहे जिथे प्रत्येकी काही अंतर्गत स्ट्रक्चर असते. फाइल रेकॉर्डचा क्रम आहे या कल्पनेचे केंद्रबिंदू म्हणजे रीड (Read) ऑपरेशन रेकॉर्ड परत करते आणि राइट (Write) ऑपरेशन फक्त रेकॉर्ड जोडते.
3. **फाइल स्ट्रक्चर 3:** आता Fig. 5.1(c) मध्ये तुम्हाला दिसणाऱ्या फाईलच्या शेवटच्या स्ट्रक्चरमध्ये, फाईलमध्ये मुळात रेकॉर्ड्सचा एक ट्री (Tree) असतो, सर्व लांबी समान असणे आवश्यक नाही. प्रत्येक रेकॉर्डमध्ये एका

फिक्स्ड स्थितीत एक की (Key) फील्ड असते आणि विशिष्ट की (Key) जलद शोधण्यासाठी ट्री (Tree) फील्डवर साठवली जाते.

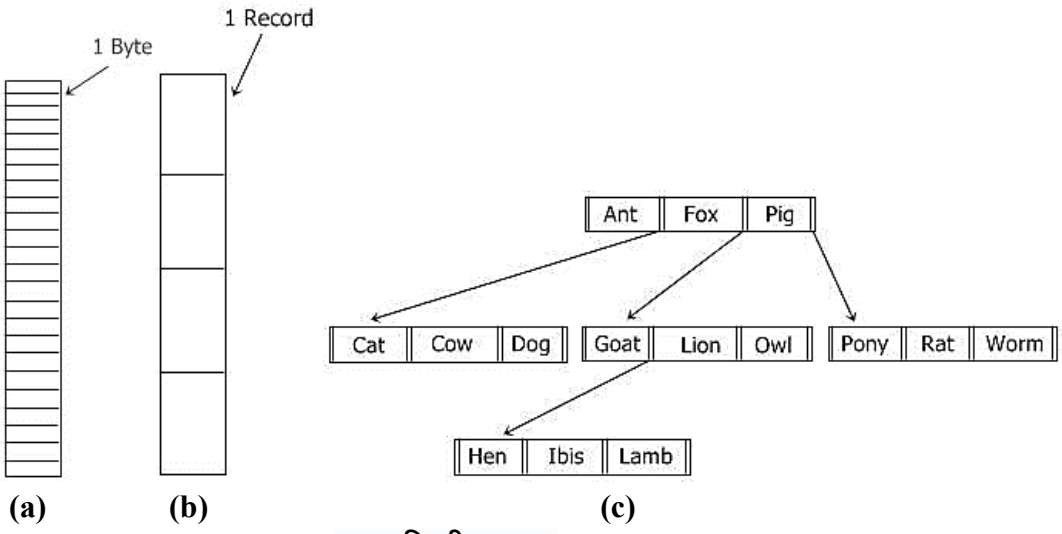


Fig. 5.1: फाइल सिस्टीम स्ट्रक्चर (File System Structure)

5.2 फाइल ऍक्सेस पद्धत (File Access method)

फाइल ऍक्सेस मेकॅनिझम म्हणजे फाइलच्या रेकॉर्डमध्ये प्रवेश कसा करता येतो याचा संदर्भ. फाइल्सला ऍक्सेस करण्याचे अनेक मार्ग आहेत –

1. सिक्वेन्शिअल ऍक्सेस (Sequential access)
2. डायरेक्ट/रँडम ऍक्सेस (Direct/Random access)
3. इंडेक्सड सिक्वेन्शिअल ऍक्सेस (Indexed sequential access)

5.2.1 सिक्वेन्शिअल ऍक्सेस (Sequential access)

सिक्वेन्शिअल ऍक्सेस म्हणजे ज्यामध्ये रेकॉर्डमध्ये काही क्रमाने ऍक्सेस केला जातो, म्हणजेच, फाइलमधील माहिती Fig.5.2 मध्ये दाखवल्याप्रमाणे एकामागून एक रेकॉर्ड क्रमाने प्रक्रिया केली जाते. ही ऍक्सेस पद्धत सर्वात प्रिमिटिव्ह आहे. सिक्वेन्शिअल ऍक्सेस ची कल्पना चुंबकीय टेप मॉडेलवर आधारित आहे जी एक सिक्वेन्शिअल ऍक्सेस डिवाइस आहे. सिक्वेन्शिअल ऍक्सेस पद्धत सर्वोत्तम आहे कारण फाइलमधील बहुतेक रेकॉर्ड प्रक्रिया करायच्या असतात जसे कि ट्रांझ्याक्शन फायली.

उदाहरण: कंपायलर (Compiler) सहसा या पद्धतीने फायलीं ऍक्सेस करतात.

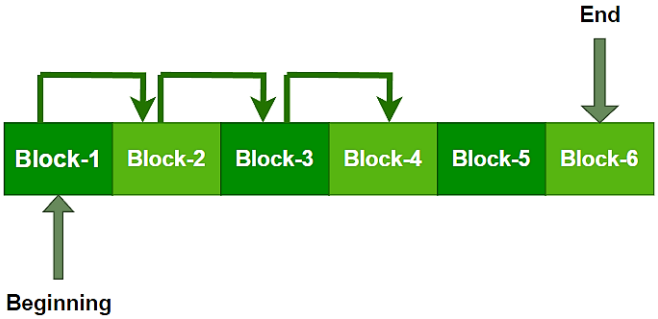


Fig. 5.2: सिक्वेन्शिअल ऍक्सेस (Sequential Access)

थोडक्यात:

- एका रेकॉर्डनंतर दुसऱ्या रेकॉर्डवर डेटा ऍक्सेस करणे हा एक ऑर्डर असते.
- रीड (Read) कमांडमुळे पॉइंटर एकामागून एक पुढे सरकतो.
- राइट (Write) कमांड रेकॉर्डसाठी जागा वाटप करते आणि पॉइंटरला नवीन फाईलच्या शेवटी हलवते.
- टेपसाठी अशी पद्धत वाजवी आहे.

सिक्वेन्शिअल ऍक्सेसचे फायदे (Advantages of sequential access)

1. प्रोग्राम करणे सोपे आणि डिझाइन करणे सोपे आहे.
2. स्टोरेज स्पेससाठी सिक्वेन्शिअल फाइलचा सर्वोत्तम वापर होतो.

सिक्वेन्शिअल ऍक्सेसचे तोटे (Disadvantages of sequential access)

1. सिक्वेन्शिअल फाइल ही वेळखाऊ प्रक्रिया आहे.
2. त्यात उच्च डेटा रिडंडन्सी आहे.
3. रँडम सर्चिंग शक्य नाही.

5.2.2 डायरेक्ट/रँडम ऍक्सेस (Direct/Random Access)

कधीकधी फाईलमधील प्रत्येक रेकॉर्डवर प्रक्रिया करणे आवश्यक नसते. सर्व रेकॉर्ड मेमरीमध्ये ज्या क्रमाने आहेत त्या क्रमाने प्रक्रिया करणे आवश्यक नसते. अशा सर्व प्रकरणांमध्ये, Fig. 5.3 मध्ये दाखवल्याप्रमाणे डायरेक्ट ऍक्सेस वापरला जातो. डिस्क हे डायरेक्ट ऍक्सेस डिव्हाइस आहे जे आपल्याला कोणत्याही फाईल ब्लॉकच्या रँडम ऍक्सेसची विश्वासार्हता देते.फाईलमध्ये, फिजिकल ब्लॉक्स आणि त्या ब्लॉक्सच्या रेकॉर्ड्सचा संग्रह असतो.

उदाहरण: डेटाबेस बहुतेकदा या प्रकारचे असतात कारण ते क्वेरी (Query) प्रक्रियेस परवानगी देतात ज्यामध्ये मोठ्या प्रमाणात माहिती त्वरित ऍक्सेस समाविष्ट असतो. सर्व आरक्षण प्रणाली (Reservation System) या श्रेणीत येतात.

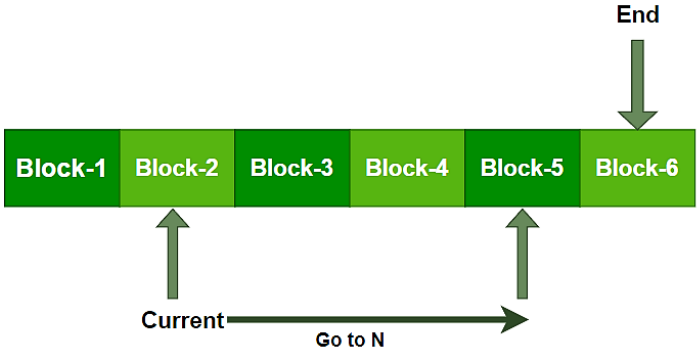


Fig. 5.3: डायरेक्ट/रँडम ऍक्सेस (Direct/Random Access)

थोडक्यात:

- ही पद्धत डिस्कसाठी उपयुक्त आहे.
- फाइल ब्लॉक्स किंवा रेकॉर्ड्सच्या सिक्वेन्स क्रमाने पाहिली जाते.
- कोणते ब्लॉक्स वाचायचे/लिहायचे यावर कोणतेही बंधन नाही, ते कोणत्याही क्रमाने करता येते.
- यूजर्स आता "पुढील वाचा" ऐवजी "पुढील n" म्हणतो.
- "n" हा फाईलच्या सुरुवातीशी संबंधित एक क्रमांक आहे, परिपूर्ण फिझिकल डिस्क लोकेशन्सशी संबंधित नाही.

फायदे (Advantages)

1. डायरेक्ट ऍक्सेस फाइल ऑनलाइन रेल्वे आरक्षण प्रणालीसारख्या ऑनलाइन ट्रांझ्याक्शन प्रोसेसिंग सिस्टिम (OLTP) मध्ये मदत करते.
2. डायरेक्ट ऍक्सेस फाइलमध्ये, रेकॉर्डचे वर्गीकरण आवश्यक नसते.
3. ते इच्छित रेकॉर्ड त्वरित ऍक्सेस करते.
4. ते अनेक फाइल्स जलद अपडेट करते.
5. रेकॉर्ड अलोकेशनवर त्याचे चांगले नियंत्रण असते.

तोटे (Disadvantages):

1. डायरेक्ट ऍक्सेस फाइल बॅकअप सुविधा प्रदान करत नाही.
2. सिक्वेन्शिअल फाइलच्या तुलनेत त्यात कमी स्टोरेज स्पेस असते.

5.2.3 इंडेक्सड सिक्वेन्शिअल ऍक्सेस (Indexed sequential access)

इंडेक्स सिक्वेन्शिअल एक्सेस मेथड ही डायरेक्ट एक्सेस मेथडची एक सुधारित आवृत्ती आहे. ही सिक्वेन्शिअल एक्सेस आणि डायरेक्ट एक्सेस या दोन्हींचे एक प्रकारचे संयोजन आहे. या पद्धतीची मुख्य कल्पना म्हणजे प्रथम फाइल थेट एक्सेस करणे आणि नंतर ती सिक्वेन्शिअल एक्सेस करते. या एक्सेस मेथडमध्ये, Fig. 5.4 मध्ये दाखवल्याप्रमाणे इंडेक्स राखणे आवश्यक आहे. इंडेक्स म्हणजे ब्लॉकला पॉइंटर करणे दुसरे काहीही नाही. इंडेक्सचा डायरेक्ट एक्सेस फाईलमधील रेकॉर्ड एक्सेस करण्यासाठी केला जातो. या एक्सेसमधून मिळणारी माहिती फाईल एक्सेस करण्यासाठी वापरली जाते. कधीकधी इंडेक्स खूप मोठे असतात. म्हणून इंडेक्सच्या या सर्व हाइरार्किची देखभाल करण्यासाठी असे तयार केले जाते ज्यामध्ये एका इंडेक्सचा डायरेक्ट एक्सेस दुसऱ्या इंडेक्स एक्सेसची माहिती देतो. ते सिक्वेन्शिअल एक्सेसच्या वर बांधले जाते. ते फाइल्स एक्सेस करताना पॉइंटर नियंत्रित करण्यासाठी इंडेक्स वापरते.

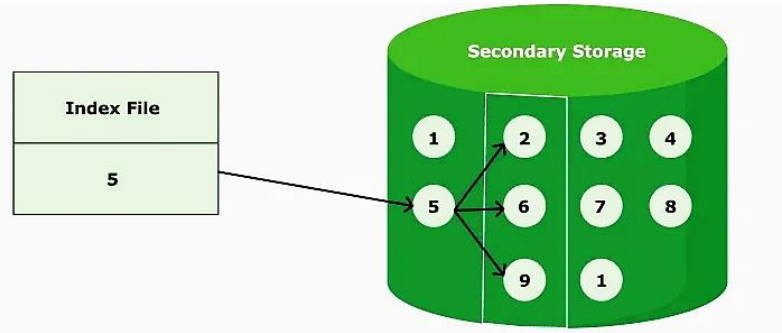


Fig. 5.4: इंडेक्सड सिक्वेन्शिअल ऍक्सेस (Index Sequential Access)

फायदे (Advantages)

1. इंडेक्सड सिक्वेन्शिअल ऍक्सेस फाइलमध्ये, सिक्वेन्शिअल फाइल आणि रँडम फाइल ऍक्सेस शक्य आहे.
2. जर इंडेक्स टेबल योग्यरित्या व्यवस्थित केली असेल तर ते रेकॉर्डमध्ये खूप जलद ऍक्सेस करते.
3. रेकॉर्ड फाइलच्या मध्यभागी इन्सर्ट केले जाऊ शकतात.
4. सिक्वेन्शिअल आणि रँडम प्रोसेसिंगसाठी ते जलद ऍक्सेस प्रदान करते.
5. हे सिक्वेन्शिअल सर्चची डिग्री कमी करते.

तोटे (Disadvantages)

1. इंडेक्सड सिक्वेन्शिअल ऍक्सेस फाइलला युनिक की (Key) आणि नियतकालिक पुनर्रचना आवश्यक असते.
2. डेटा ऍक्सेस किंवा पुनर्प्राप्तीसाठी इंडेक्स शोधण्यासाठी इंडेक्सड सिक्वेन्शिअल ऍक्सेस फाइलला जास्त वेळ लागतो.
3. त्यासाठी अधिक स्टोरेज स्पेस आवश्यक असते.
4. ते एक्स्पेन्सिव्ह असते कारण त्यासाठी विशेष सॉफ्टवेअर आवश्यक असते
5. इतर फाइल ऑर्गनायझेशनच्या तुलनेत ते स्टोरेज स्पेसच्या वापरात कमी कार्यक्षम आहे.

5.3 फाइल अलोकेशन पद्धत (File Allocation Method)

ऑपरेटिंग सिस्टमद्वारे फाइल्सना डिस्क स्पेस वाटप केले जातात. ऑपरेटिंग सिस्टम फाइल्सना डिस्क स्पेस वाटप करण्यासाठी खालील तीन मुख्य मार्गांचा वापर करतात.

1. कॉन्टिग्युअस अलोकेशन (Contiguous Allocation)
2. लिंक्ड अलोकेशन (Linked Allocation)
3. इंडेक्स्ड अलोकेशन (Indexed Allocation)

5.3.1 कॉन्टिग्युअस अलोकेशन (Contiguous Allocation)

या टेक्निक मध्ये, प्रत्येक फाईल डिस्कवर ब्लॉक्सचा एक कॉन्टिग्युअस संच व्यापते. उदाहरणार्थ, जर एखाद्या फाईलला **n** ब्लॉक्सची आवश्यकता असेल आणि त्याला सुरुवातीचे लोकेशन म्हणून ब्लॉक **b** दिला असेल, तर फाईलला नियुक्त केलेले ब्लॉक्स असे असतील: **b, b+1, b+2,……b+n-1.** याचा अर्थ असा की सुरुवातीचा ब्लॉक ॲड्रेस आणि फाइलची लांबी (आवश्यक ब्लॉक्सच्या बाबतीत) पाहता, आपण फाइलने व्यापलेले ब्लॉक्स निश्चित करू शकतो. कॉन्टिग्युअस अलोकेशन असलेल्या फाईलसाठी डिरेक्टरी एंट्रीमध्ये

1. सुरुवातीचा ब्लॉकचा ॲड्रेस
2. अलोकेट केलेल्या भागाची लांबी.

Fig. 5.5 मधील फाइल '**mail**' ब्लॉक **19** पासून लांबी = **6** ब्लॉक्ससह सुरू होते. म्हणून, ती **19, 20, 21, 22, 23, 24** ब्लॉक्स व्यापते.

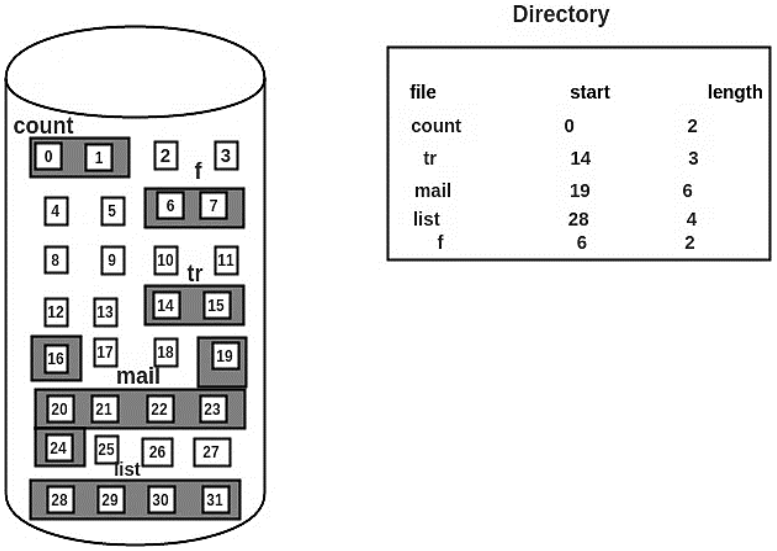


Fig. 5.5: कॉन्टिग्युअस अलोकेशन (Contiguous Allocation)

प्रत्येक फाईल डिस्कवर एका संलग्न ॲड्रेस ची जागा व्यापते; नियुक्त डिस्क ॲड्रेस लिनिअर क्रमाने असतो. म्हणून इम्प्लिमेंट करणे सोपे आहे आणि एक्सटर्नल फ्रॅगमेंटेशन ही या प्रकारच्या वाटप टेक्निकमध्ये एक प्रमुख समस्या आहे.

फायदे (Advantages):

1. याद्वारे सिक्वेन्शिअल आणि डायरेक्ट ॲक्सेस दोन्ही समर्थित आहेत. डायरेक्ट ॲक्सेससाठी, ब्लॉक b पासून सुरू होणाऱ्या फाइलच्या kth ब्लॉकचा ॲड्रेस (b+k) म्हणून सहजपणे मिळवता येतो.
2. हे अत्यंत जलद आहे कारण फाइल ब्लॉक्सच्या कॉन्टिग्युअस अलोकेशनमुळे शोधांची संख्या कमी आहे.

तोटे (Disadvantages):

1. ही पद्धत इंटर्नल आणि एक्सटर्नल दोन्ही फ्रॅगमेंटेशनने ग्रस्त आहे. यामुळे मेमरी वापराच्या बाबतीत ती अकार्यक्षम बनते.
2. फाइल आकार वाढवणे कठीण आहे कारण ते एका विशिष्ट ठिकाणी कॉन्टिग्युअस मेमरीच्या उपलब्धतेवर अवलंबून असते.

5.3.2 लिंक्ड अलोकेशन (Linked Allocation)

या टेक्निकमध्ये, प्रत्येक फाइल डिस्क ब्लॉक्सची एक लिंक्ड लिस्ट असते जी कॉन्टिग्युअस असण्याची गरज नाही. डिस्क ब्लॉक्स डिस्कवर कुठेही विखुरलेले असू शकतात. डिरेक्टरी एंट्रीमध्ये सुरुवातीच्या आणि शेवटच्या फाइल ब्लॉकला एक पॉइंटर असतो. प्रत्येक ब्लॉकमध्ये फाइलने व्यापलेल्या पुढील ब्लॉकला पॉइंटर असतो. Fig. 5.6 मधील फाइल 'जीप' मध्ये ब्लॉक्स कसे रँडमली वितरित केले जातात ते दाखवले आहे. शेवटच्या ब्लॉक (**25**) मध्ये -**1** आहे जो नल पॉइंटर दर्शवितो आणि इतर कोणत्याही ब्लॉकला पॉइंट करत नाही. प्रत्येक फाईलमध्ये डिस्क ब्लॉक्सच्या लिंक्सची यादी असते आणि डिरेक्टरीमध्ये फाईलच्या पहिल्या ब्लॉकची लिंक/पॉइंटर असते. म्हणून, कोणतेही एक्सटर्नल फ्रॅगमेंटेशन नसते, सिक्वेन्शिअल ॲक्सेस फाइलमध्ये प्रभावीपणे वापरले जाते परंतु डायरेक्ट ॲक्सेस फाइलच्या बाबतीत अकार्यक्षम असते.

फायदे (Advantages):

1. फाइल आकार निर्दिष्ट करण्याची आवश्यकता नाही.
2. एक्सटर्नल फ्रॅगमेंटेशन नाही.

तोटे (Disadvantages):

1. ते सिक्वेन्शिअल ॲक्सेस कार्यक्षमतेने करते आणि डायरेक्ट प्रवेशासाठी नाही.
2. प्रत्येक ब्लॉकमध्ये एक पॉइंटर असतो, ज्यामुळे जागा वाया जाते.

3. ब्लॉक सर्वत्र विखुरलेले असतात आणि मोठ्या संख्येने डिस्क शोध आवश्यक असू शकते.
4. विश्वसनीयता: जर पॉइंटर हरवला किंवा खराब झाला तर काय**?**

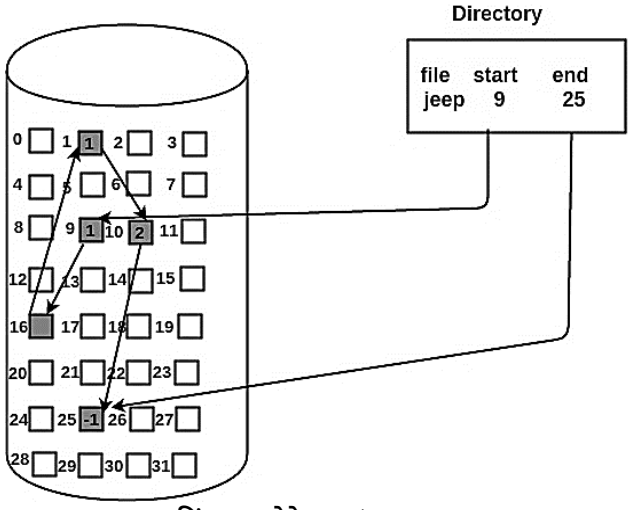


Fig. 5.6: लिंक्ड अलोकेशन (Linked Allocation)

5.3.3 इंडेक्स्ड अलोकेशन (Indexed Allocation)

या टेक्निक मध्ये**,** इंडेक्स ब्लॉक म्हणून ओळखल्या जाणाऱ्या एका विशेष ब्लॉकमध्ये फाइलने व्यापलेल्या सर्व ब्लॉक्सचे पॉइंटर्स असतात. प्रत्येक फाइलचा स्वतःचा इंडेक्स ब्लॉक असतो. इंडेक्स ब्लॉकमधील ith एंट्रीमध्ये ith फाइल ब्लॉकचा डिस्क ॲड्रेस असतो. Fig. 5.7 मध्ये दाखवल्याप्रमाणे डायरेक्टरी एंट्रीमध्ये इंडेक्स ब्लॉकचा ॲड्रेस असतो. कॉन्टिग्युअस आणि लिंक्ड अलोकेशनच्या समस्यांचे निराकरण करते. सर्व पॉइंटर्ससह एक इंडेक्स ब्लॉक तयार केला जातो. प्रत्येक फाईलचा स्वतःचा इंडेक्स ब्लॉक असतो जो फाईलने व्यापलेल्या डिस्क स्पेसचे अड्रेसेस साठवतो. डिरेक्टरीमध्ये फाइल्सच्या इंडेक्स ब्लॉकचे अड्रेसेस असतात.

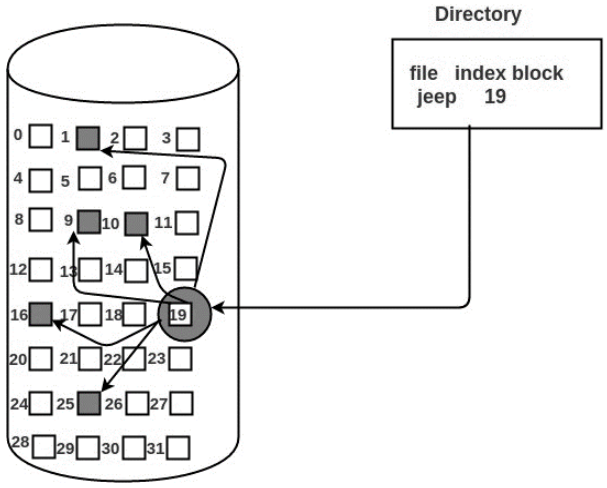


Fig. 5.7: इंडेक्स्ड अलोकेशन (Indexed Allocation)

फायदे (Advantages):

1. हे फाईलने व्यापलेल्या ब्लॉक्सला थेट ॲक्सेस करण्यास समर्थन देते आणि त्यामुळे फाइल ब्लॉक्समध्ये जलद ॲक्सेस प्रदान करते.
2. हे एक्सटर्नल फ्रॅगमेंटेशनच्या समस्येवर मात करते.

तोटे (Disadvantages):

1. इंडेक्स्ड अलोकेशनसाठी पॉइंटर ओव्हरहेड लिंक्ड अलोकेशनपेक्षा जास्त आहे.
2. खूप लहान फाइल्ससाठी, उदाहरणार्थ ज्या फाइल्स फक्त 2-3 ब्लॉक्स वाढवतात, इंडेक्स्ड अलोकेशन पॉइंटर्ससाठी एक संपूर्ण ब्लॉक (इंडेक्स ब्लॉक) ठेवेल जे मेमरी वापराच्या बाबतीत अकार्यक्षम आहे. तथापि, लिंक्ड अलोकेशनमध्ये आपण प्रति ब्लॉक फक्त 1 पॉइंटरची जागा गमावतो.

5.4 डिरेक्टरी (Directory)

ऑपरेटिंग सिस्टममधील डिरेक्टरी ही एक विशेष प्रकारची कंटेनर/फोल्डर असते जी अनेक फाइल्स आणि फोल्डर्स व्यवस्थितपणे साठवण्यासाठी वापरली जाते. किंवा आपण असे म्हणू शकतो की ते एका कंटेनर म्हणून काम करते ज्यामध्ये फाइल्स आणि इतर डिरेक्टरीज (सब डिरेक्टरी) दोन्ही ठेवता येतात, ज्यामुळे एक हाइरार्किकल ऑर्गनिझेशन होते. डिरेक्टरीज यूजर्सना आणि प्रोग्राम्सना हार्ड ड्राइव्ह किंवा एसएसडी (SSD) सारख्या स्टोरेज डिव्हाइसेसवर डेटा कार्यक्षमतेने ऍक्सेस, स्टोअर आणि ऑर्गनाईझ करण्यास मदत करतात. डिरेक्टरीजची हाइरार्किकल रचना रूट डिरेक्टरीपासून सुरू होते, ज्यामुळे फाइल्स शोधणे आणि फाइल सिस्टममध्ये सुव्यवस्था राखणे सोपे होते.

5.4.1 डिरेक्टरी स्टक्चर (Directory Structure)

ऑपरेटिंग सिस्टममधील डिरेक्टरीचे स्ट्रक्चर म्हणजे फाइल सिस्टममध्ये फाइल्स आणि डिरेक्टरीज कशा प्रकारे ऑर्गनाईझ केल्या जातात याचा संदर्भ. ते स्टोरेज डिव्हाइसवर डेटा कसा संग्रहित केला जातो, ऍक्सेस केला जातो आणि मॅनेज केला जातो हे परिभाषित करते. चांगल्या प्रकारे डिझाइन केलेली डिरेक्टरी स्ट्रक्चर कार्यक्षम फाइल ऑर्गनायजेशन, सोपे नेव्हिगेशन आणि वाढत्या डेटासाठी स्केलेबिलिटी सुनिश्चित करते. ऑपरेटिंग सिस्टम विविध प्रकारच्या डिरेक्टरी स्ट्रक्चर प्रदान करते आणि त्या आहेत:

1. सिंगल लेव्हल डिरेक्टरी (Single-Level Directory)
2. टू-लेव्हल डिरेक्टरी (Two-Level Directory)
3. ट्री स्ट्रक्चर डिरेक्टरी (Tree Structure Directory)

1. सिंगल लेव्हल डिरेक्टरी (Single-Level Directory)

एका ऑपरेटिंग सिस्टममध्ये सिंगल लेव्हल डिरेक्टरी ही सर्वात सोपी प्रकारची डिरेक्टरी रचना आहे. या स्ट्रक्चरमध्ये, सर्व फायली एकाच डिरेक्टरीमध्ये संग्रहित केल्या जातात आणि Fig. 5.8 मध्ये दाखवल्याप्रमाणे कोणत्याही सब डिरेक्टरी नसतात. यूजर्स आणि सिस्टम दोघेही समान डिरेक्टरी जागा शेअर करतात. तथापि, जेव्हा फायलींची संख्या वाढते किंवा जेव्हा सिस्टममध्ये एकापेक्षा जास्त यूजर्स असतात तेव्हा सिंगल लेव्हल डिरेक्टरीत लक्षणीय मर्यादा असते. सर्व फायली एकाच डिरेक्टरीत असल्याने, त्यांचे युनिक नाव असणे आवश्यक आहे. जर दोन यूजर्स त्यांच्या डेटासेट चाचणीला कॉल करतात, तर युनिक नाव नियमाचे उल्लंघन होते.

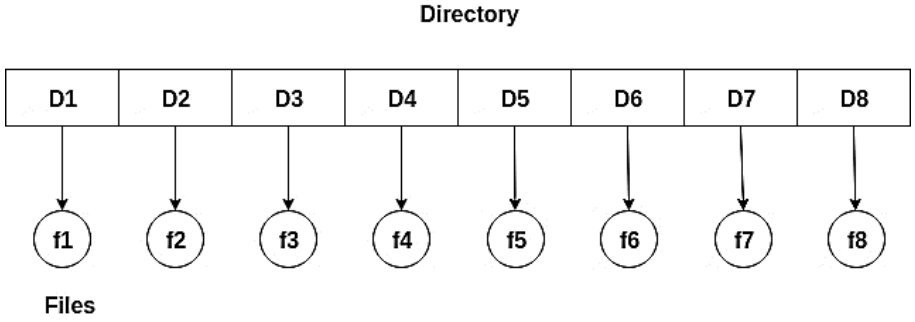


Fig.5.8: सिंगल लेव्हल डिरेक्टरी (Single-Level Directory)

फायदे (Advantages)

1. ही एकच डिरेक्टरी असल्याने, तिचे इम्प्लिमेंट करणे खूप सोपे असते.
2. जर फाइल्स आकाराने लहान असतील तर शोध जलद होईल.
3. अशा डिरेक्टरी स्ट्रक्चर फाइल तयार करणे, शोधणे, हटवणे, अपडेट करणे यासारखी कामे खूप सोपी होतात.

तोटे (Disadvantages)

1. दोन फाइल्सना एकच नाव असू शकत नाही म्हणून नाव कॉन्फ्लिक्ट होण्याची शक्यता असते.
2. जर डिरेक्टरी मोठी असेल तर शोधण्यात वेळ लागेल.
3. यामध्ये एकाच प्रकारच्या फाइल्स एकत्र गटबद्ध करता येत नाहीत.

2. टू-लेव्हल डिरेक्टरी (Two-Level Directory)

सिंगल लेव्हल डिरेक्टरीमुळे वेगवेगळ्या यूजर्स मध्ये फाइल्सच्या नावांचा गोंधळ होतो, म्हणून या समस्येचे निराकरण म्हणजे प्रत्येक यूजर्ससाठी स्वतंत्र डिरेक्टरी तयार करणे.

टू-लेव्हल डिरेक्टरी स्ट्रक्चरमध्ये, Fig. 5.9 मध्ये दाखवल्याप्रमाणे प्रत्येक यूजर्स स्वतःची युजर फाइल्स टू-लेव्हल डिरेक्टरी (UFD) असते. UFD मध्ये समान स्ट्रक्चर्स असतात, परंतु प्रत्येकामध्ये फक्त एकाच युजरच्या फाइल्सची यादी असते. जेव्हा जेव्हा नवीन युजर आयडी लॉग इन केला जातो तेव्हा सिस्टमची मास्टर फाइल डायरेक्टरी (MFD) शोधते. यूजरनेम किंवा अकॉउंट नंबर, आणि प्रत्येक एंट्री त्या यूजर साठी UFD कडे पॉईंट करते MFD. येथे MFDs यूजरनेम किंवा अकॉउंट नंबरद्वारे इंडेक्सड केले जातात जे यूजरच्या प्रत्येक एंट्री पॉइंटसह UFD कडे पॉईंट केले जातात. टू-लेव्हल डिरेक्टरी स्ट्रक्चर फायली शोधणे अधिक सोपे होते कारण फक्त एकच यूजरची यादी असते, जी प्रत्येक फाइलसाठी /User-name/directory-name/ सारखे पाथनेम असणे आवश्यक असते जे येथे देखील परिभाषित केले असते.

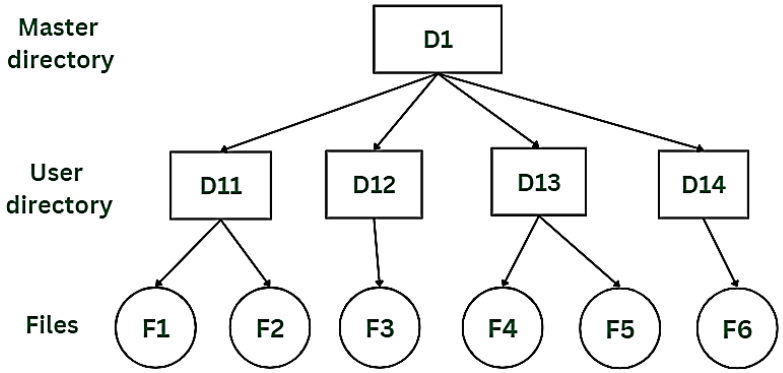


Fig. 5.9: टू-लेव्हल डिरेक्टरी (Two-Level Directory)

फायदे (Advantages)

1. आपण /**User-name/directory-name/** सारखा पूर्ण पाथ देऊ शकतो.
2. वेगवेगळ्या यूजर्सच्या फाइल नावाप्रमाणेच डिरेक्टरी देखील असू शकते.
3. पाथनेम आणियूजर्स ग्रुपिंग मुळे फाइल्स शोधणे सोपे होते.

तोटे (Disadvantages):

1. यूजरला इतर यूजर्ससोबत फाइल्स शेअर करण्याची परवानगी नसते.
2. तरीही ते फारसे स्केलेबल नाही**,** एकाच प्रकारच्या दोन फाइल्स एकाच यूजरमध्ये एकत्र केल्या जाऊ शकत नाहीत.

3. ट्री स्ट्रक्चर डिरेक्टरी (Tree-structured directory)

आपल्या वैयक्तिक संगणकांमध्ये ऑपरेटिंग सिस्टमची ट्री डायरेक्टरी स्ट्रक्चर सर्वात जास्त वापरली जाते. यूजर्स फाइल्स आणि सब-डिरेक्टरीज देखील तयार करू शकतो, जी मागील डिरेक्टरी स्ट्रक्चर्समध्ये एक गैरसोय होती. ही डिरेक्टरी स्ट्रक्चर उलट्या ट्रीच्या रूपात असते, जिथे रूट डिरेक्टरी शिखरावर असते. या रूटमध्ये प्रत्येक यूजर्ससाठी सर्व डिरेक्टरीज असतात. यूजर्स सबडिरेक्टरीज तयार करू शकतात आणि त्यांच्या डिरेक्टरीमध्ये फाइल्स देखील साठवू शकतात. युजरला रूट डिरेक्टरीमधील डेटा ऍक्सेस करता येत नाही आणि तो तो बदलू शकत नाही. या डिरेक्टरीमध्ये देखील युजरला इतर यूजर्सचा डिरेक्टरीजमध्ये ऍक्सेस नाही. ट्री डिरेक्टरीची स्ट्रक्चर खाली दिली आहे जी प्रत्येक यूजर्सच्या डिरेक्टरीमध्ये फाइल्स आणि सबडिरेक्टरीज कसे आहेत हे दर्शवते. Fig. 5.10 मध्ये दाखवल्याप्रमाणे ट्री स्ट्रक्चर ही सर्वात सामान्य डिरेक्टरी स्ट्रक्चर आहे. ट्रीला रूट डिरेक्टरी असते आणि सिस्टममधील प्रत्येक फाइलचा एक युनिक मार्ग असतो.

फायदे (Advantages)

1. खूप जनरलाईज**,** कारण पूर्ण पाथचे नाव दिले जाऊ शकते.
2. खूप स्केलेबल**,** नावचे कोलीजन होण्याची शक्यता कमी असते.
3. शोधणे खूप सोपे होते**,** आपण अबसोल्यूट मार्ग तसेच रिलेटिव्ह पाथ दोन्ही वापरू शकतो.

तोटे (Disadvantages)

1. प्रत्येक फाइल हाइरार्किकल मॉडेलमध्ये बसत नाही**;** फाइल्स अनेक डिरेक्टरीज मध्ये जतन केल्या जाऊ शकतात.
2. आपण फाइल्स शेअर करू शकत नाही.
3. ते अकार्यक्षम आहे**,** कारण फाइल ऍक्सेस करणे अनेक डिरेक्टरीज मध्ये जाऊ शकते.

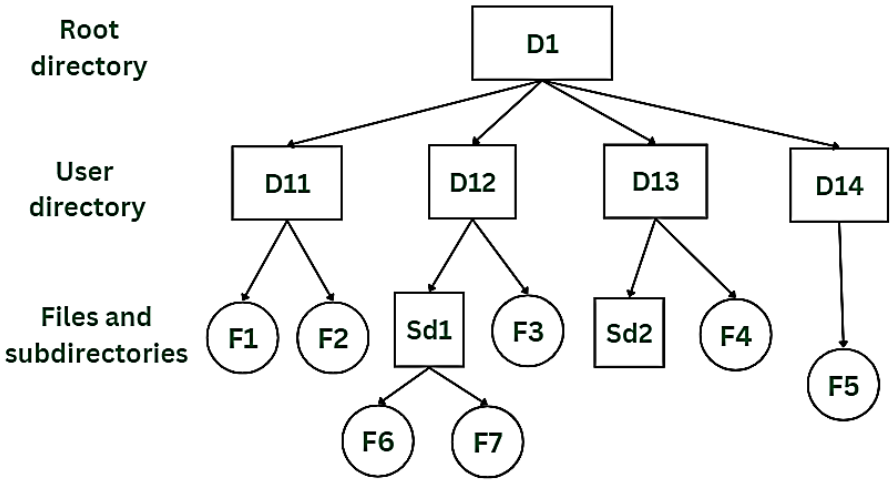


Fig. 5.10: ट्री स्ट्रक्चर डिरेक्टरी (Tree-structured directory)

References:

1. Operating System: A Concept-Based Approach by Dhananjay M. Dhamdhere, McGraw Hill Education 3rd edition, ISBN: 978-1259005589
2. Operating Systems : Internals and Design Principles by William Stallings, Pearson Education 9th Edition, ISBN: 978-9352866717
3. Linux The Complete Reference by Richard Petersen, McGraw Hill, 6th edition, ISBN: 978-0071492478
4. Linux command line and shell scripting by Richard Blum, Wiley India, ISBN: 978-1118983843
5. Operating System Concepts by Abraham Silberschatz and James Peterson, Wiley India, ISBN: 9781119454083

Websites:

1. https://www.tutorialspoint.com/computer_concepts/computer_concepts_file_directory_management.htm
2. https://www.geeksforgeeks.org/file-access-methods-in-operating-system/
3. https://www.javatpoint.com/os-file-access-methods
4. https://www.tutorialspoint.com/file-access-method
5. https://www.tutorialspoint.com/operating_system/os_memory_management.htm
6. https://www.geeksforgeeks.org/structures-of-directory-in-operating-system/
7. https://www.scaler.com/topics/directory-structure-in-os-part2/

E-learning material (Video lectures)

1. https://www.youtube.com/watch?v=Lhr48wCtvKU – Concept of files and directory
2. https://www.youtube.com/watch?v=ywWPul7VAwM – Directory Structures
3. https://www.youtube.com/watch?v=ZN-baY3x85o – Swapping
4. https://www.youtube.com/watch?v=tI3MACfFfFc – File allocation method
5. https://www.youtube.com/watch?v=S6lLRz7SQUw –File allocation method
6. https://www.youtube.com/watch?v=eK_mEHG20PA – Directory Structure
7. https://www.youtube.com/watch?v=Ja9EeFVid6I – Disk Organization
8. https://www.youtube.com/watch?v=Iqs5G-Uz9gg – Disk Structure