DEAD LOCK

डेडलॉक और सिस्टम मॉडल की सरल व्याख्या (Operating System में)

डेडलॉक एक ऐसी स्थिति है जिसमें एक या अधिक प्रक्रियाएँ (processes) हमेशा के लिए रुकी रहती हैं क्योंकि वे आवश्यक संसाधनों को प्राप्त नहीं कर पा रही हैं। इसे बेहतर समझने के लिए हम **सिस्टम मॉडल** को तीन चरणों में समझ सकते हैं:

1) **रिक्वेस्ट (Request)**

जब कोई प्रक्रिया (process) किसी संसाधन (resource) की जरूरत महसूस करती है, तो वह ऑपरेटिंग सिस्टम से उस संसाधन को **मांगती (request करती)** है। उदाहरण के लिए, अगर कोई प्रोग्राम प्रिंटर का उपयोग करना चाहता है, तो वह पहले प्रिंटर संसाधन की रिक्वेस्ट भेजेगा।

2) **यूज़ (Use)**

अगर ऑपरेटिंग सिस्टम वह संसाधन उपलब्ध करवा देता है, तो प्रक्रिया (process) उसे **इस्तेमाल (use) करना शुरू** कर देती है। इस दौरान, कोई अन्य प्रक्रिया उस संसाधन को तब तक नहीं ले सकती जब तक पहली प्रक्रिया उसका इस्तेमाल कर रही है।

3) **रिलीज़ (Release)**

जब प्रक्रिया अपना काम पूरा कर लेती है, तो वह संसाधन को **छोड़ (release) देती** है ताकि कोई अन्य प्रक्रिया उसे उपयोग कर सके।

डेडलॉक कब होता है?

अगर कोई प्रक्रिया संसाधन को छोड़ने से पहले किसी अन्य संसाधन की मांग कर रही हो, और दूसरी प्रक्रिया भी इसी स्थिति में फंसी हो, तो कोई भी प्रक्रिया आगे नहीं बढ़ सकती–यही **डेडलॉक** कहलाता है।

डेडलॉक की परिभाषा (Definition of Deadlock)

डेडलॉक ऑपरेटिंग सिस्टम में **एक ऐसी स्थिति** होती है जब **दो या अधिक प्रक्रियाएँ (processes) संसाधनों (resources) का इंतजार कर रही होती हैं लेकिन कोई भी संसाधन को छोड़ नहीं रहा होता**, जिससे **सभी प्रक्रियाएँ स्थायी रूप से अटक जाती हैं** और आगे नहीं बढ़ सकतीं।

डेडलॉक का परिदृश्य (Deadlock Scenario)

मान लीजिए कि दो प्रक्रियाएँ **P1 और P2** हैं:

- **P1** को संसाधन **R1** मिला हुआ है, लेकिन उसे संसाधन **R2** की भी जरूरत है।
- **P2** को संसाधन **R2** मिला हुआ है, लेकिन उसे संसाधन **R1** की जरूरत है।
- **P1 और P2 दोनों एक-दूसरे का इंतजार कर रही हैं**, लेकिन कोई भी संसाधन छोड़ नहीं रही है।

इस स्थिति में **कोई भी प्रक्रिया आगे नहीं बढ़ सकती**, जिससे **डेडलॉक हो जाता है**।

डेडलॉक के लिए आवश्यक शर्तें (Necessary Conditions for Deadlock)

डेडलॉक **चार शर्तों (conditions)** के पूरा होने पर होता है:

1️⃣ Mutual Exclusion (म्यूचुअल एक्सक्लूशन)

- संसाधन को **एक समय में केवल एक प्रक्रिया** ही उपयोग कर सकती है।
- कोई और प्रक्रिया उसी समय उस संसाधन का उपयोग नहीं कर सकती।

2️⃣ Hold and Wait (होल्ड एंड वेट)

- कोई प्रक्रिया **पहले से कुछ संसाधन रखे हुए होती है** और उसे **नए संसाधनों की जरूरत होती है**।

- वह पुराने संसाधनों को छोड़े बिना नए संसाधनों का इंतजार करती है।

3️⃣ **No Preemption (नो प्री-एम्प्शन)**

- कोई भी प्रक्रिया **किसी अन्य प्रक्रिया से जबरदस्ती संसाधन नहीं ले सकती**।
- संसाधन को **स्वेच्छा से ही छोड़ा जा सकता है**।

4️⃣ **Circular Wait (सर्कुलर वेट)**

- प्रक्रियाएँ **एक चक्र (circle) में संसाधनों के लिए इंतजार** कर रही होती हैं।
- उदाहरण: **P1 को P2 के संसाधन की जरूरत है, P2 को P3 के संसाधन की जरूरत है, और P3 को P1 के संसाधन की जरूरत है।**

जब ये **चारों शर्तें एक साथ पूरी होती हैं**, तब **डेडलॉक की समस्या उत्पन्न होती है**।

Dead Lock Prevention :-

1) म्यूचुअल एक्सक्लूज़न रोकथाम (Mutual Exclusion Prevention):
अगर संसाधन (resource) को केवल एक प्रक्रिया ही इस्तेमाल कर सकती है, तो यह डेडलॉक का कारण बन सकता है। इसको रोकने के लिए हम कोशिश करते हैं कि संसाधन साझा (share) किए जा सकें या ऐसी तकनीक अपनाई जाए जिससे एक्सक्लूज़न की जरूरत ना पड़े।

2) होल्ड एंड वेट रोकथाम (Hold and Wait Prevention):
डेडलॉक तब होता है जब कोई प्रक्रिया एक संसाधन को पकड़कर रखती है और नए संसाधनों का इंतजार करती रहती है। इसे रोकने के लिए, प्रक्रिया को तभी संसाधन दिए जाते हैं जब उसके पास सभी ज़रूरी संसाधन उपलब्ध हों। या फिर उसे सभी संसाधन छोड़ने के बाद ही नए संसाधन मांगने की अनुमति दी जाती है।

3) नॉन-प्रीएम्पशन रोकथाम (Non-Preemption Prevention):
अगर कोई प्रक्रिया संसाधन ले चुकी है, तो उसे जबरदस्ती वापस नहीं लिया जाता। लेकिन अगर हम ऐसी व्यवस्था बनाएं जिसमें ज़रूरत पड़ने पर संसाधन किसी दूसरी प्रक्रिया को दे दिए जाएं (forcefully लेने की सुविधा हो), तो डेडलॉक को रोका जा सकता है।

4) सर्कुलर वेट रोकथाम (Circular Wait Prevention):
डेडलॉक तब होता है जब प्रक्रियाएँ एक चक्र (circle) में संसाधन के लिए इंतजार कर रही होती हैं। इसे रोकने के लिए हम संसाधनों को एक क्रम (order) में नंबर देते हैं और प्रक्रियाओं को उसी क्रम में संसाधन लेने की अनुमति देते हैं। इससे सर्कुलर डिपेंडेंसी नहीं बनती।

Memory management Unit :-

1. मेमोरी मैनेजमेंट क्या है?

मेमोरी मैनेजमेंट वह प्रक्रिया है जिसके तहत ऑपरेटिंग सिस्टम आपके कंप्यूटर की RAM (रैंडम एक्सेस मेमोरी) का सही ढंग से उपयोग सुनिश्चित करता है। इसके मुख्य कार्य हैं:

- **प्रोग्राम्स को जगह देना:** जब कोई प्रोग्राम चलता है, तो OS RAM में उसे एक उपयुक्त स्थान देता है।
- **मल्टीटास्किंग को सपोर्ट करना:** कई प्रोग्राम्स एक साथ चलते हैं। OS यह तय करता है कि कौन सा प्रोग्राम किस हिस्से में चलेगा।
- **मेमोरी का पुन: उपयोग:** जब कोई प्रोग्राम बंद हो जाता है, तो OS उस जगह को खाली करके अगले प्रोग्राम के लिए उपलब्ध कराता है।
- **सुरक्षा:** मेमोरी की सुरक्षा सुनिश्चित करना ताकि एक प्रोग्राम की जानकारी दूसरे प्रोग्राम से प्रभावित न हो।

इस प्रकार, मेमोरी मैनेजमेंट से कंप्यूटर तेज़ी से और सुचारू रूप से काम करता है।

2. लॉजिकल एड्रेस और फिजिकल एड्रेस

जब कोई प्रोग्राम चलता है, तो उसे मेमोरी में डेटा रखने या पढ़ने के लिए एड्रेस (पता) की ज़रुरत होती है। ये एड्रेस दो प्रकार के होते हैं:

(a) लॉजिकल एड्रेस (Logical Address)

- **क्या है:**
  लॉजिकल एड्रेस वह पता है जिसे CPU द्वारा प्रोग्राम के दौरान जनरेट किया जाता है। इसे वर्चुअल एड्रेस भी कहा जाता है।
- **कैसे काम करता है:**
  जब प्रोग्राम कोई डेटा पढ़ने या लिखने का अनुरोध करता है, तो वह एक लॉजिकल एड्रेस भेजता है। इस एड्रेस को उस समय तक कंप्यूटर की असली मेमोरी (RAM) का पता नहीं होता। OS के मेमोरी मैनेजमेंट यूनिट (MMU) द्वारा इस एड्रेस को असली (फिजिकल) एड्रेस में बदला जाता है।
- **सरल शब्दों में:**
  आप इसे एक नक़ली पता समझ सकते हैं जो प्रोग्राम के लिए काम करता है, लेकिन असल में इसे बाद में बदलकर असली मेमोरी के पते से जोड़ा जाता है।

(b) फिजिकल एड्रेस (Physical Address)

- **क्या है:**
  फिजिकल एड्रेस असली RAM में रिकॉर्ड होने वाला पता होता है जहाँ डेटा स्टोर होता है। यह हार्डवेयर (RAM) द्वारा इस्तेमाल किया जाता है।
- **कैसे काम करता है:**
  जब कोई प्रोग्राम अपना डेटा मेमोरी में सेव करता है, तो अंत में वह डेटा

RAM में किसी निश्चित स्थान पर रखा जाता है। यह असली पता ही फिजिकल एड्रेस कहलाता है। इसे MMU द्वारा लॉजिकल एड्रेस से मैप किया जाता है।

- **सरल शब्दों में:**
  यह वही असली पता है जहाँ आपका डेटा भौतिक रूप से (RAM में) रखा जाता है।

SWAPPIN IN MEMORY THROW OPERATING SYSTEM

स्वैपिंग (Swapping) ऑपरेटिंग सिस्टम की एक महत्वपूर्ण मेमोरी मैनेजमेंट तकनीक है, जो तब काम में आती है जब RAM (रैंडम एक्सेस मेमोरी) में पर्याप्त खाली जगह नहीं बचती। सरल शब्दों में, स्वैपिंग वह प्रक्रिया है जिसमें कम उपयोग में आए या निष्क्रिय प्रोग्राम के हिस्सों (या पूरे प्रोसेस) को अस्थायी रूप से हार्ड डिस्क पर स्थित एक विशेष क्षेत्र, जिसे स्वैप स्पेस (Swap Space) या स्वैप फाइल कहा जाता है, में स्थानांतरित कर दिया जाता है। इससे RAM में खाली जगह बन जाती है, जिसका उपयोग अन्य सक्रिय प्रोग्रामों द्वारा किया जा सकता है।

स्वैपिंग की प्रक्रिया कैसे काम करती है?

1. **प्रोसेस का चयन:**
   जब सिस्टम में मेमोरी की कमी होती है, तो OS ऐसे प्रोसेस का चयन करता है जो वर्तमान में सक्रिय नहीं हैं या जिनका उपयोग कम हो रहा है।

2. **स्वैप आउट (Swap Out):**
   चयनित प्रोसेस के मेमोरी में स्टोर सभी डेटा को हार्ड डिस्क पर बने स्वैप स्पेस में कॉपी कर दिया जाता है। इस प्रक्रिया को स्वैप आउट कहते हैं। इससे RAM में पर्याप्त खाली जगह बन जाती है।

3. **RAM में स्थान की उपलब्धता:**
   नए या सक्रिय प्रोसेस को RAM में लोड करने के लिए इस खाली जगह का उपयोग किया जाता है।

4. **स्वैप इन (Swap In):**
   जब पहले स्वैप किए गए प्रोसेस को फिर से सक्रियता की आवश्यकता होती है या उसमें से कुछ डेटा एक्सेस करना होता है, तब OS उस प्रोसेस के डेटा को स्वैप स्पेस से लेकर RAM में वापस लाता है। इस प्रक्रिया को स्वैप इन कहा जाता है।

स्वैपिंग के फायदे और नुकसान

- **फायदे:**
  - **अधिक प्रोसेस का एक साथ निष्पादन:**
    सीमित RAM के बावजूद, कई प्रोग्राम एक साथ चल सकते हैं क्योंकि कम उपयोग में आने वाला डेटा अस्थायी रूप से डिस्क पर भेज दिया जाता है।
  - **मेमोरी का कुशल प्रबंधन:**
    OS RAM का अधिकतम उपयोग सुनिश्चित करता है और आवश्यकतानुसार मेमोरी आवंटित करता है।
- **नुकसान:**
  - **धीमी पहुँच:**
    हार्ड डिस्क की गति RAM की तुलना में बहुत कम होती है। जब बार-बार स्वैपिंग होती है, तो सिस्टम की प्रदर्शन (performance) में गिरावट आ सकती है, जिसे थ्रेशिंग (Thrashing) कहते हैं।
  - **अतिरिक्त I/O संचालन:**
    स्वैपिंग की प्रक्रिया में डिस्क पर लगातार पढ़ाई और लिखाई के

कारण इनपुट/आउटपुट (I/O) संचालन बढ़ जाते हैं, जो कि सिस्टम को धीमा कर सकते हैं।

1) **फिक्स्ड-साइज पार्टिशन स्कीम (Fixed-Size Partition Scheme)**

इस स्कीम में मेमोरी को **समान आकार के कई पार्टिशन्स** में विभाजित किया जाता है। हर पार्टिशन का आकार पहले से तय होता है और इसमें केवल उतनी ही जगह दी जाती है।

लक्षण:

- प्रत्येक पार्टिशन का साइज **फिक्स्ड** होता है।
- एक **प्रोसेस को सिर्फ एक पार्टिशन** असाइन किया जाता है।
- यदि कोई प्रोसेस छोटे आकार का है और पार्टिशन बड़ा है, तो **इंटरनल फ्रैगमेंटेशन** (Internal Fragmentation) होती है यानी अतिरिक्त जगह बेकार जाती है।

फायदे:

- इम्प्लीमेंटेशन **सरल** होता है।
- मेमोरी मैनेजमेंट **आसान** होता है।

नुकसान:

- **फिक्स्ड पार्टिशन** होने की वजह से **इंटरनल फ्रैगमेंटेशन** बढ़ती है।
- यदि कोई बड़ा प्रोसेस है और सभी पार्टिशन छोटे हैं, तो वह मेमोरी में **लोड नहीं हो सकता** (External Fragmentation)।

2) **वेरिएबल-साइज पार्टिशन स्कीम (Variable-Size Partition Scheme)**

इस स्कीम में मेमोरी को **डायनामिक रूप से प्रोसेस की जरूरत के अनुसार विभाजित** किया जाता है। यह फिक्स्ड पार्टिशन की तुलना में अधिक लचीलापन प्रदान करता है।

लक्षण:

- हर प्रोसेस को उसकी **जरूरत के अनुसार** मेमोरी दी जाती है।
- **बचे हुए स्पेस को पुन: उपयोग** किया जा सकता है।
- **फिक्स्ड साइज की कोई बाध्यता नहीं** होती।

फायदे:

- **इंटरनल फ्रैगमेंटेशन नहीं** होती क्योंकि प्रोसेस के अनुसार मेमोरी एलोकेट होती है।
- अधिक **फ्लेक्सिबिलिटी** मिलती है।

नुकसान:

- **एक्सटर्नल फ्रैगमेंटेशन** (External Fragmentation) हो सकती है, जहाँ छोटे-छोटे खाली स्पेस बनते हैं जिनका उपयोग करना कठिन हो सकता है।
- मेमोरी मैनेजमेंट **थोड़ा जटिल** होता है।

Paging एक मेमोरी मैनेजमेंट तकनीक है जो ऑपरेटिंग सिस्टम द्वारा **प्रोसेस और मेमोरी के कुशल उपयोग** को सुनिश्चित करने के लिए इस्तेमाल की जाती है। यह **फ्रैगमेंटेशन की समस्या को हल करने** में मदद करता है।

Paging की अवधारणा:

Paging में, **मेमोरी को छोटे-छोटे फिक्स्ड-साइज ब्लॉक्स** में विभाजित किया जाता है, जिन्हें **फ्रेम (Frames)** कहते हैं। इसी तरह, प्रोसेस को भी **समान आकार के ब्लॉक्स** में विभाजित किया जाता है, जिन्हें **पेज (Pages)** कहते हैं।

जब कोई प्रोसेस मेमोरी में लोड होता है, तो उसके पेज **एविलेबल फ्रेम्स** में मैप किए जाते हैं। इससे मेमोरी का **अधिकतम उपयोग** होता है और **कंटिग्युअस (लगातार) मेमोरी स्पेस की जरूरत नहीं** पड़ती।

Paging का कार्य करने का तरीका:

1. **फिक्स्ड-साइज फ्रेम्स**: मेमोरी को कई छोटे-छोटे ब्लॉक्स (Frames) में विभाजित किया जाता है।
2. **फिक्स्ड-साइज पेजेज**: प्रोसेस को भी समान आकार के ब्लॉक्स (Pages) में विभाजित किया जाता है।
3. **पेज टेबल**: हर प्रोसेस के लिए एक **पेज टेबल** बनाई जाती है, जो बताती है कि कौन सा पेज किस फ्रेम में स्टोर किया गया है।
4. **पेज ट्रांसलेशन**: जब प्रोसेस को किसी डेटा की जरूरत होती है, तो **लॉजिकल एड्रेस** को **फिजिकल एड्रेस** में बदला जाता है, जिससे सही फ्रेम से डेटा एक्सेस किया जा सके।