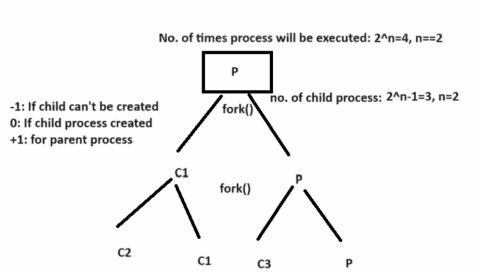
Day 3

Date : 29 aug 2024

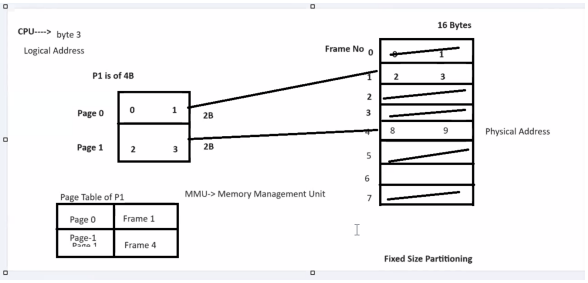
Content Discussed Day 3:

C Thing : Formula ⇒ How many time process will be executed , where n is number of fork used .



Memory Management ⇒ Paging

1. Divide the whole process into number of sectio just before entering into ram , then why not to just divide the process into number of pages and load them in hard drive (secondary memory) and loading it according to the need .
2. Divide the process into number of page when ⇒ at the very first time when program is in hard drive and load them according to that we will load them in ram which accordance to size available .
3. PHYSICAL MEMORY(RAM) IS DIVIDED INTO **FRAMES** AND LOGICAL (HARDD RIVE) IS **PAGES** .
4. Only the demanded page will get loaded not the whole process at once.
5. **Only external fragmentation can be avoided using paging .**
6. **Pages will be generated in hard drive only , not at runtime in ram.**
7. Process is divided into pages , and memory us divided into frames.
8. To support paging should size of page should be equal to size of frames.
9. Size of pages is equivalent and same size and same size of frames .
10. Every frame in ram is of same size .
11. Due to that no internal or external fragmentations .
12. We need primary as well secondary memory for paging to happen.
13. PC are byte addressable means byte by byte and ram is also divided into bytes.
14. Then there is comes CPU which want to read these bytes .
15. Request Mapping : To Map the address there is proper unit to do this ⇒ **Memory management Unit ⇒ Logical address into physical address .**



Demand Paging :

- Demand Paging: any page of a process gets loaded into the main memory only after
  requesting by that process i.e. on demand and hence referred as demand paging.
- The page which is never requested never gets loaded into the main memory and hence it also called as pure demand paging.
- If a process is requesting for any page and if that page is not exists in the main memory, then it is referred as **page fault.**
- As the size of process may be bigger than size of main memory itself, and in a system multiple processes are running at a time, hence no. of pages are more than no. of frames, so there are quite good chances that all frames becomes full, and in this case if any process is requesting for a page which does not exists in the main memory at that time there is need to remove any one page from the main memory so that into that free frame requested page will get loaded.
- So there is need to decide which page should get removed and requested page gets replaced in that frame, to do this there are certain algorithms referred as page replacement algorithms .

16. MMU generates a page table to do so per process ⇒ proper mapping of pages and frames .
17. What is the need? The CPU demanding the byte may not be the same as the memory present, that's why MMU does this work.

Page Replacement Algorithms :

1. FIFO Page Replacement: page which was inserted first gets replaced by the
requested page.
2
. Optimal Page Replacement: page which will not get used in a near future gets
replaced by the requested page
3.
LRU(Least Recently Used) Page Replacement: least recently used page gets
replaced by the requested page.
4
. LFU(Least Frequently Used) Page Replacement: least frequently used page gets
replaced by the requested page.
5.
MFU(Most Frequently Used) Page Replacement: most frequently used page gets
replaced by the requested page.

Virtual Memory Management:

Virtual memory / Swap : (16gb if ram is 8gb)
1.
As we seen an OS does memory management for completing an execution of multiple submitted processes
at once.
2. An OS also able to complete an execution of such a process having size bigger than size of main memory itself, and to achieve this an OS manages swap area memory as well with main memory and hence it is referred as virtual memory management.
3. As an OS manages such a memory which is physically not a main memory and hence it is referred as virtual memory management.
4. Virtual memory management can be implemented by using **Paging + Swapping.**
in this technique for a process to complete its execution it is not mandatory it must exists wholly in a main memory, even its part is their into the main memory then execution of such process can be completed part
by part.
5 .Big size process is divided into pages and when a process is requesting for memory few pages gets loaded
into the main memory and few pages can be kept into the swap area, and as per the request pages of that process can swapped in and swapped out between main memory and swap area.

This acts like an RAM but never execute any process and it is volatile like the content will vanish out when OS will turn off .

Pages which we create will also get stored in virtual memory , then it load some part into the RAM when needed .

> 💡 In FIFO Belady's Anomaly said on increasing no. of frames the hit ratio decreases .

Virtual Memory :

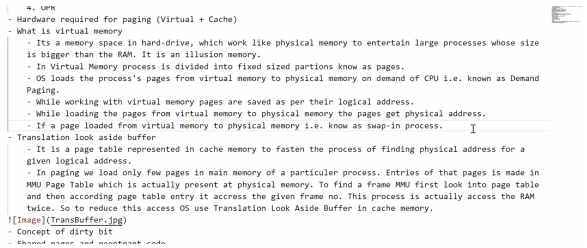
1. The process are divided into fixed size pages .
2. Pc will divide the memory into bytes .
3. The number 15 can be represented in **four bits** in the binary number system.

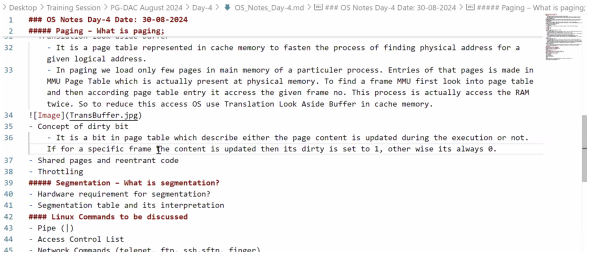
In the binary system, numbers are represented using only the digits 0 and 1 (bits). To determine how many bits are needed to represent a number, you can start with the number 1 and double it until you get a value that's greater than the number you want to represent.

> if we have logical address of 4gb , my page size is 2 byte , how many process wil be generated

> 💡 you have 1 gb logical address space and memory is byte addressable , how many address i will need ?
> In EXAM ⇒ logical address spacing

1. To write 16 we want 4 bits , then 64 bits PC means ki in pc it can hold address in 64 bits .
2. Logical addresses are very big because the hard drive is too much space , and physical memory address are small because the ram is small size .





Topic: Recording
Module: COS
Date: 29/08/24

https://us02web.zoom.us/rec/share/BZD2CXB55Xpp72J_WReKI1nrtBruPJqrfYCE0BJHwCtU7Rpp9r6Ayqi1jpMTt_bL.nam1rAMfFjkI9ucY?startTime=1724918678000