Process Scheduling Algorithm D3 Revise

29 August 2024 13:38

Process Scheduling Algorithm

1. First come first serve

   Advantage
   CPU attend every process when its term is come.

   Disadvantage
   Longer Waiting time
   (convey effect) [read more]

2. Shortest Job First

   Process with short burst time execute first

   Advantage:
   Every process executes in time.

   Disadvantage:
   Starvation - cpu will not attain larger burst time [read more]

3. Round Robin

   Advantages:
   Every process gives certain amount of time to process. This certain time or slice called quantum.

   Disadvantage: [read more]
   Contact switching - cpu after expiration of time slices of certain process cpu will save this process in harddrive for certain time (swap out) then after some time it cpu will take that process to main memory(swap in).

   efficiency

In a hour how much process will cpu execute is the throughput of cpu

Disadvantage of one algorithm gives rise to another algorithm.

Arrival - when process came in memory
Response - when cpu assigned to process
Completion - when process is done

Process creation using fork Day 3

29 August 2024 14:25

Process creation using fork

Fork()
It is a method or system call used to create child process of current process being executed.

Child process does exactly same as parent process designed for.

Example of process creation

nano Program.c

[in Program.c

```
#include<stdio.h>
Int main(){
Printf("Hello All\n");
Return 0;
}
```

]

```
gcc -o Program Program.c
./Program
nano Program.c
```

[in Program.c

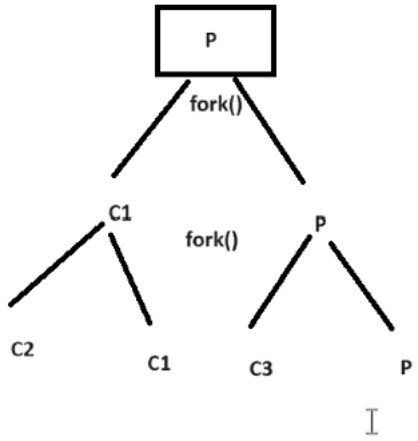
```
#include<stdio.h>
Int main(){
fork();
Printf("Hello All\n");
Return 0;
}
```

]

```
gcc -o Program Program.c
./Program
```

```
malkeet@CMKL-malkeet:~$
malkeet@CMKL-malkeet:~$
malkeet@CMKL-malkeet:~$ nano Program.c
malkeet@CMKL-malkeet:~$ gcc -o Program Program.c
malkeet@CMKL-malkeet:~$ ./Program
Hello Allmalkeet@CMKL-malkeet:~$ nano Program.c
malkeet@CMKL-malkeet:~$ gcc -o Program Program.c
malkeet@CMKL-malkeet:~$ ./Program
Hello All
malkeet@CMKL-malkeet:~$ nano Program.c
malkeet@CMKL-malkeet:~$ gcc -o Program Program.c
Program.c: In function ‘main’:
Program.c:4:1: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
    4 | fork();
      | ^~~~
malkeet@CMKL-malkeet:~$ ./Program
Hello All
Hello All
```

fork() is concept when c++, multithreading created. So fork() is used because at that time they want parent will do some tasks and child will do different task. Then eventually it gives rise to multithreading



2^n is used to calculate copy of process or number of time process will be executed.
Ex. If fork used two times then n=2 therefore copies of process are 4.
2^n-1 is for how many child processes created
Ex. n=2 then child processes are 2.

-1 if child can't be created.
0 if child process is created.
+1 for parent process.

fork-exec-wait
If one process is executing other process waits

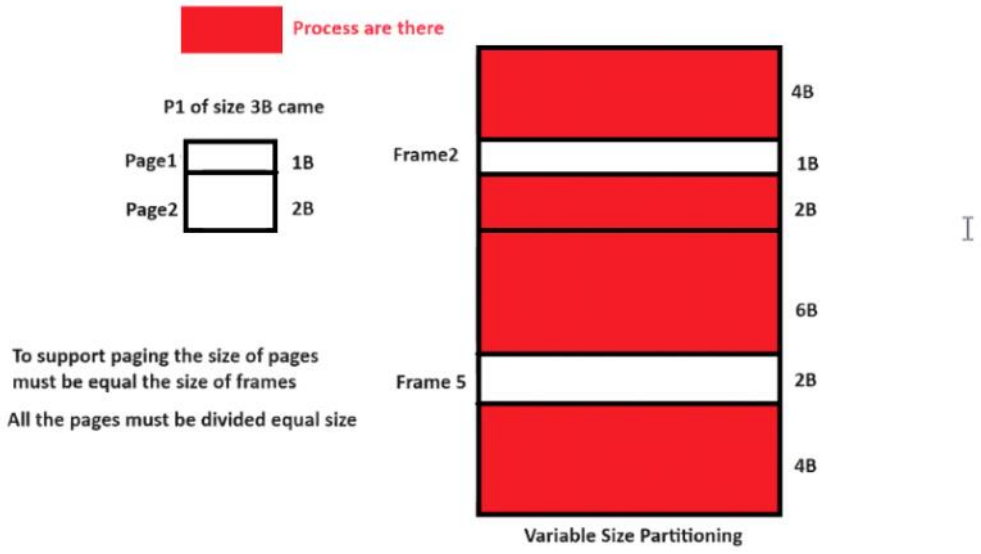
Zombie process - parent process completed and child is being executed now parent process still remain in memory until the child process is executing completely. So the parent process is become dead process. Hence parent process is Zombie.

Orphan process - parent get executed and child process is executing then child process is Orphan.

Memory Management - Paging D3

29 August 2024 14:45

Variable Size Partitioning



-Each block is not same size
-Problem occur is external fragmentation

P1 need contagious memory allocation.

If it need non contagious memory allocation then it can be fitted out.

It is not possible to divide process exactly before going to the ram. Cant divide process at run time

So divide it in hard drive is called paging
Divide process into number of pages.

Dividing the process into fixed sized pages is known as paging.

Paging is used for -

- Instead of loading entire program into main memory just load needed one.
- Paging is used to remove external fragmentation.
- it will better utilize the ram
- it will give maximum throughput and better efficiency

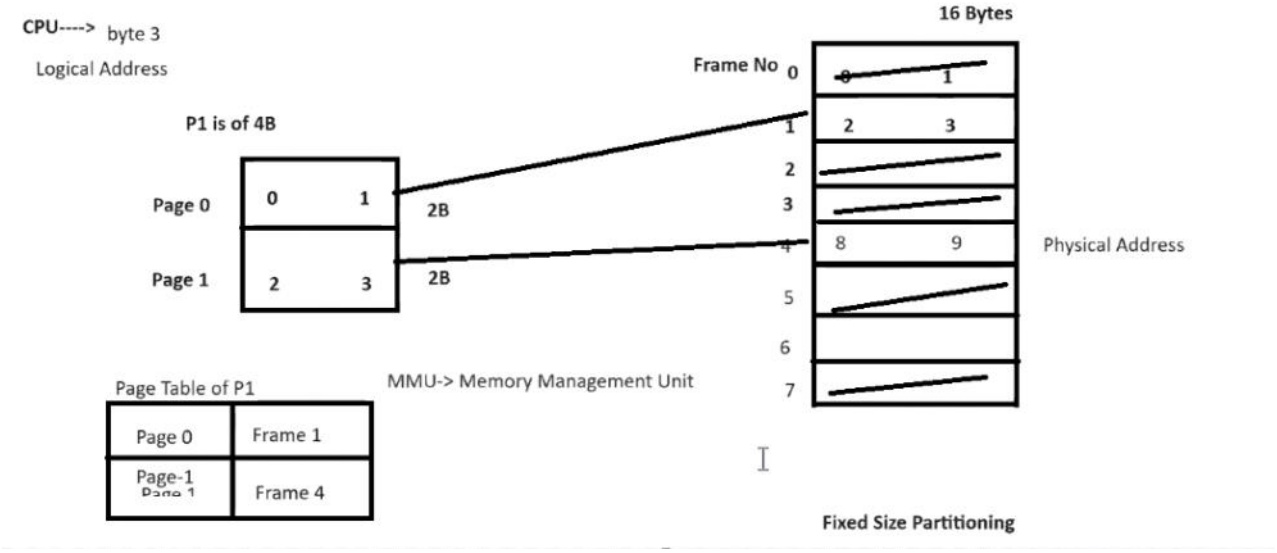
For paging:
Program divide into pages and
Ram is divided into frames

If OS is designed to support paging -
The size of frame should be equal to size of pages

All pages must be divided into equal size and frames are divided into equal size.
And due to that there is no internal and external fragmentation.
That's why paging can not done in above figure given.

Both the main and secondary memories we needed for paging.

Fixed Size Partitioning



As computer support binary language that's why indexing start from zero.

All computer systems are byte addressable. (Reading the memory according to per byte)

If we have P1 of 4 bytes. Then it will get divide into 2 pages and frames of 2 bytes per page.

Physical memory has physical address and logical memory has logical address.

Cpu generates bytes have logical address
But in Memory bytes have physical address.
To map this address there is memory management unit (MMU) .
Mapping of logical address to physical address is done by MMU.
It is done by using page table.
In page table there is proper mapping of pages and frames is done.

Technically byte 3 is at byte 9 in figure that is called as mapping.

In cpu there is logical address and in ram there is physical address. Some of the frames of ram are occupied and some are free. Hence when bytes from cpu goes into ram they get converted according to the frames that are not occupied. It is done by MMU. Hence here logical address 3 is converted to physical address 9.

All this is done by OS.

Virtual memory acts like a ram which
also helps in paging.

Demand Paging : If the page is
loaded in memory frames as per the

demand of CPU i.e. Demand paging

Page Fault : If the page is not in frame, it is known as miss of the mage and it's a page fault.

If cpu wants a page but it is not available in memory is called page fault.

Page replacement algorithm:
1. Most recently used (MRU)
2. Least Recently Used (LRU)
3. Optimal Page Replacement (ORU)

1. Most recently used (MRU)
(jo recently use hua)

P1---->	0	2	4	6	8	7	3	5	3	2
		0	2	4	6	8	7	3	5	3	2
Frame-1	0	0	2	4	6	8	7	7	7	7	7
Frame-2	3	3	3	3	3	3	3	3	3	3	2
Frame-3	5	5	5	5	5	5	5	5	5	5	5
	Initial	HIT	MISS	MISS	MISS	MISS	MISS	HIT	HIT	HIT	MISS

MISS Ration=60%
HIT= 40%

Every miss here is a page fault.
Here is Most recently used (MRU) page replacement algorithm is used. Therefore at second 0 replaced by 2 because 0 is most recently used and at last 3 is replaced by 2 because 3 is most recently used.

If the page frames are empty then load p1 in frames. For ex.
Frame 1 - 0
Frame 2 - 2
Frame 3 - 4

2. Least Recently Used (LRU)
(jo bahot pahle use hua)

P1----->	2	3	2	4	3	5	4	3	2	7	
LRU											
		2	3	2	4	3	5	4	3	2	7
Frame-1	8	8	8	8	4	4	4	4	4	4	4
Frame-2	3	3	3	3	3	3	3	3	3	3	3
Frame-3	2	2	2	2	2	2	2	2	2	2	2
Frame-4	6	6	6	6	6	6	5	5	5	5	7
	Init	HIT	HIT	HIT	MISS	HIT	MISS	HIT	HIT	HIT	MISS
					Mis %	30					
					Hit %	70					

32 bit system = 32 bit chunk
64 bit system = 8 byte chunk

4 bit = 2^4 = 16 memory chunk = 16
values can store
16 addresses can store
Like,
0000
0001
.
.
.
1111