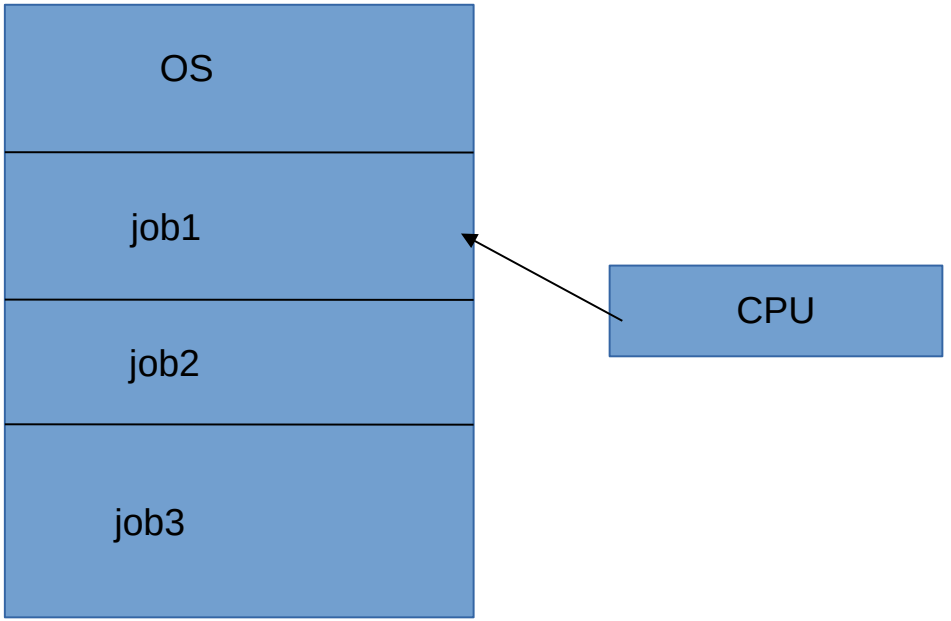
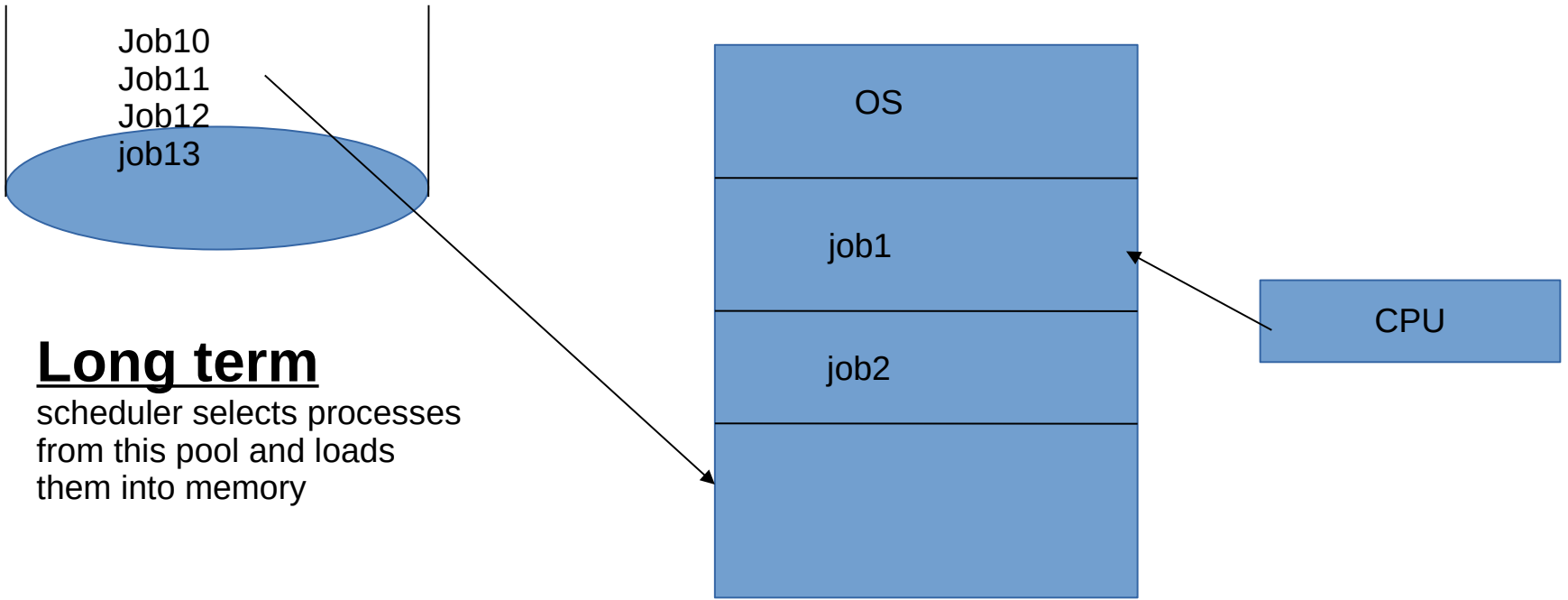
Once a process is executing one of the following events can occur

1. The process could issue an I/O request and then be placed in an I/O queue.
2. The CPU time expire(2 sec for each process).
3. The Process could create a new sub process and wait for its termination.
4. The process could be removed forcibly from the CPU as a result of an interrupt and be put back in the ready queue.

There are sometimes more processes submitted than can be executed immediately.

These processes are kept in a storage device (usually a disk) for later execution.

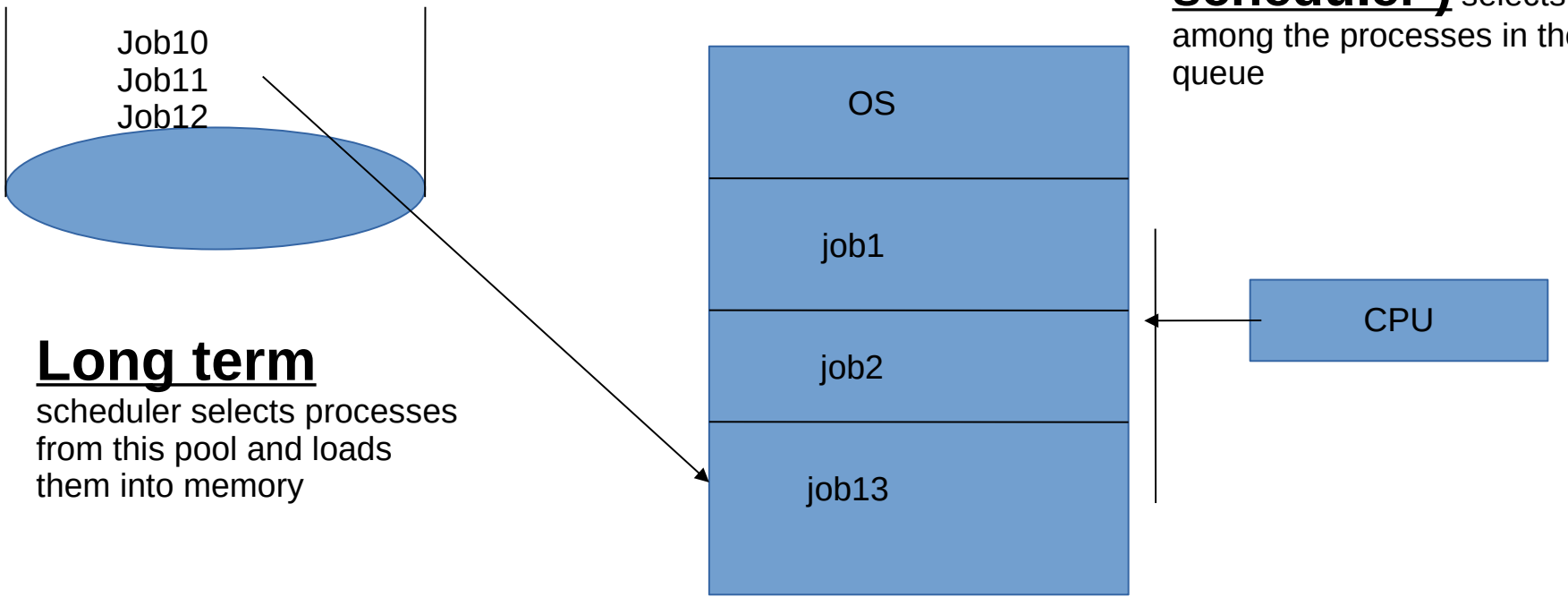




Long term

scheduler selects processes from this pool and loads them into memory

Short term Scheduler(CPU scheduler) selects from among the processes in the ready queue



Long term
scheduler selects processes from this pool and loads them into memory

. Long term scheduler executes less frequently than the short term scheduler.

. Long term scheduler controls the degree of multiprogramming.

I/O and CPU bound process

```
#include<stdio.h>
Int main()
{
    Int p;
    P=10;
    printf("\n initial value=%d",p);
    While(1)
    {
        p=p+1;
    }
}
```

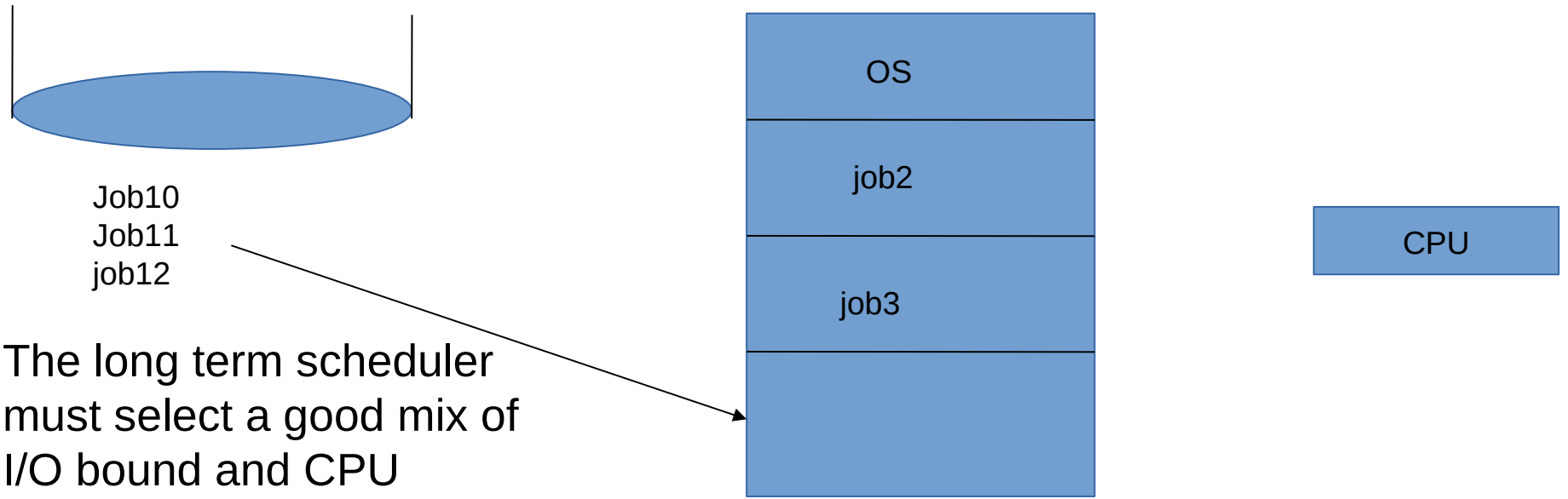
```
#include<stdio.h>
 int main()
   {
     int q;
       p=100;
          while(1)
              {
                  printf("\n hello");
              }
          }
```

- If all processes are CPU bound processes then the I/O devices sit idle.

- If all processes are I/O bound processes then the CPU sit idle.

- Utilization Problem.

The long term scheduler is responsible for solving this problem.

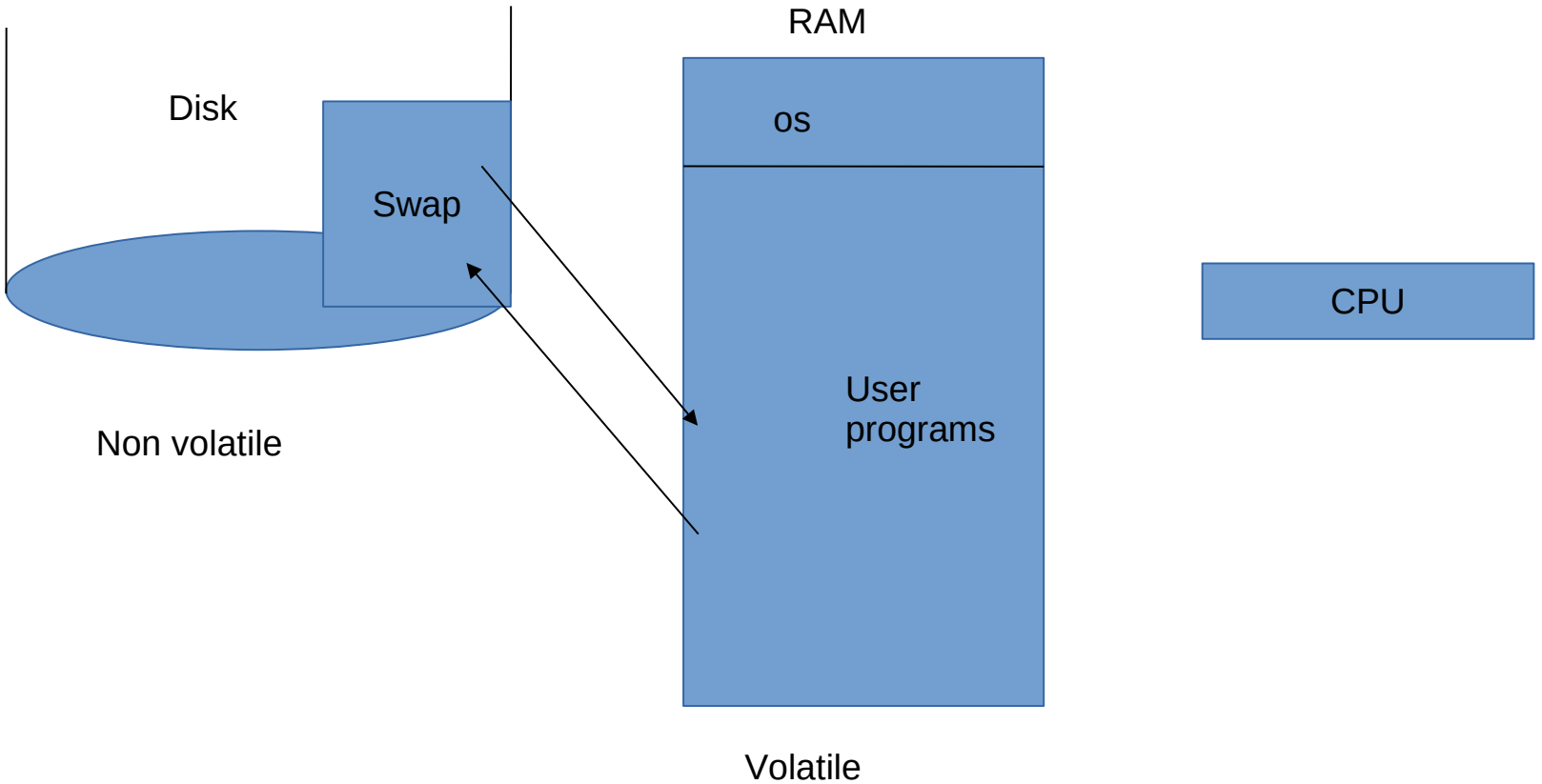


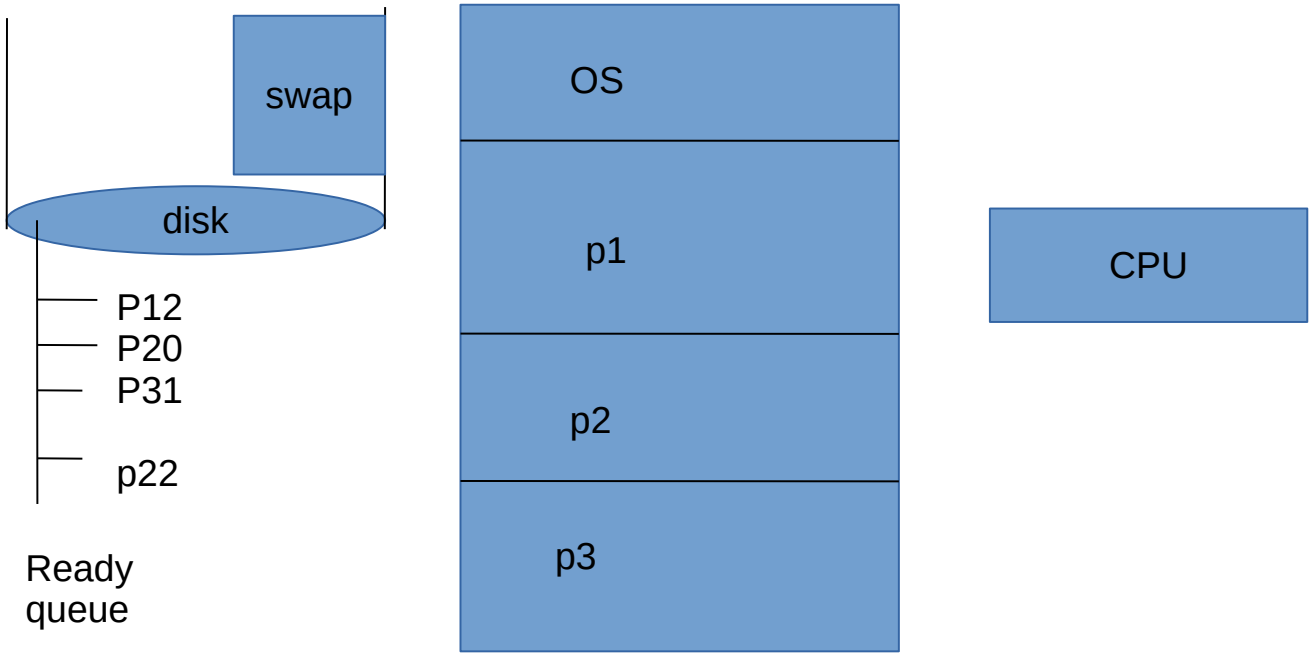
The long term scheduler must select a good mix of I/O bound and CPU bound processes.

Mid Term Scheduler

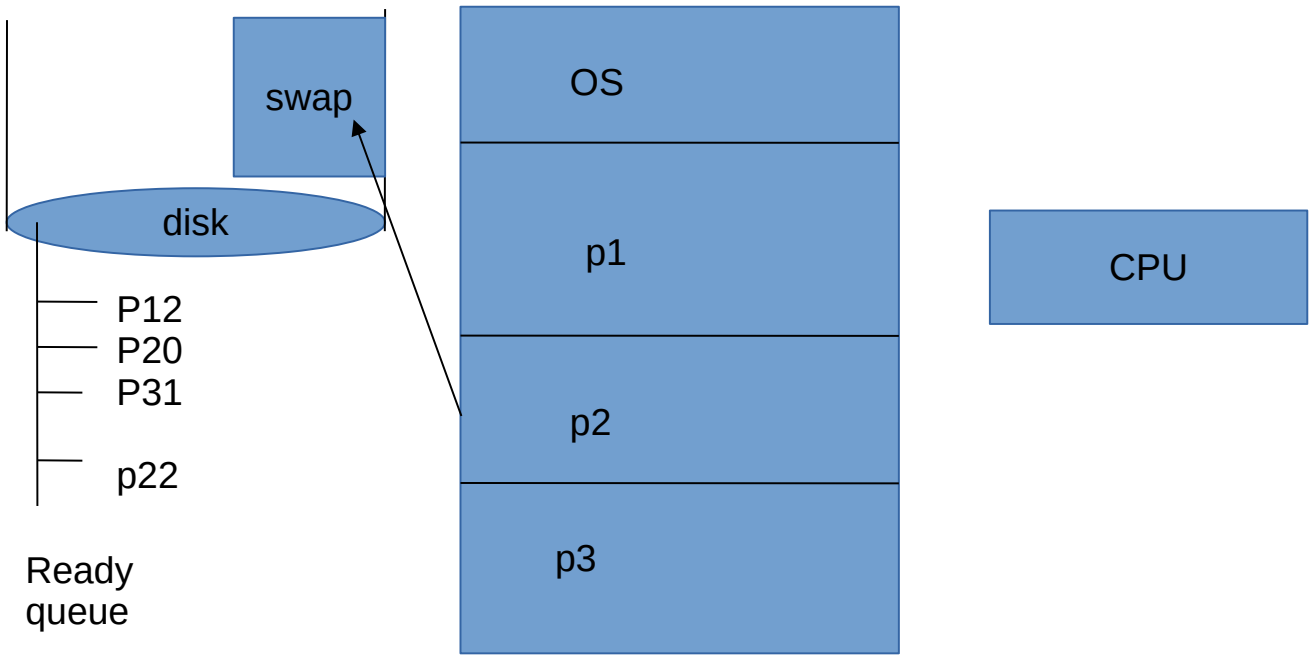
- Swapping

Hibernate






```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```

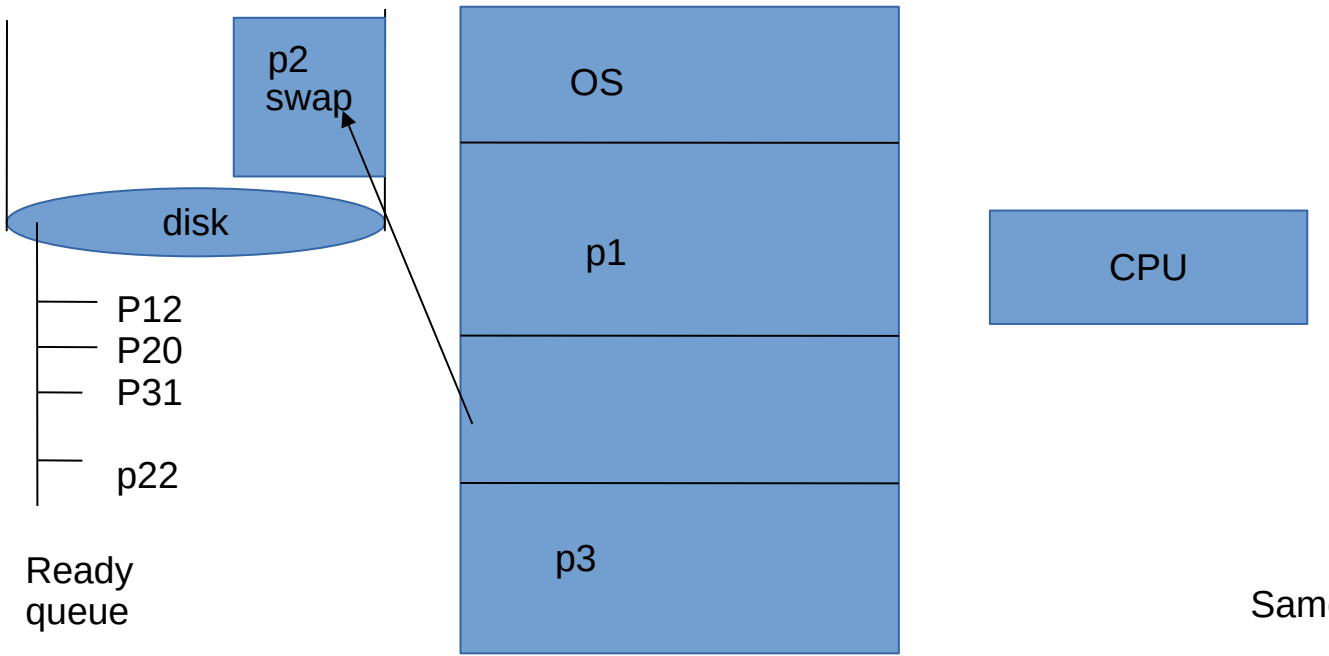



```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```

p2

When p2 is doing I/O then, is p2 required to be in main memory?

NO



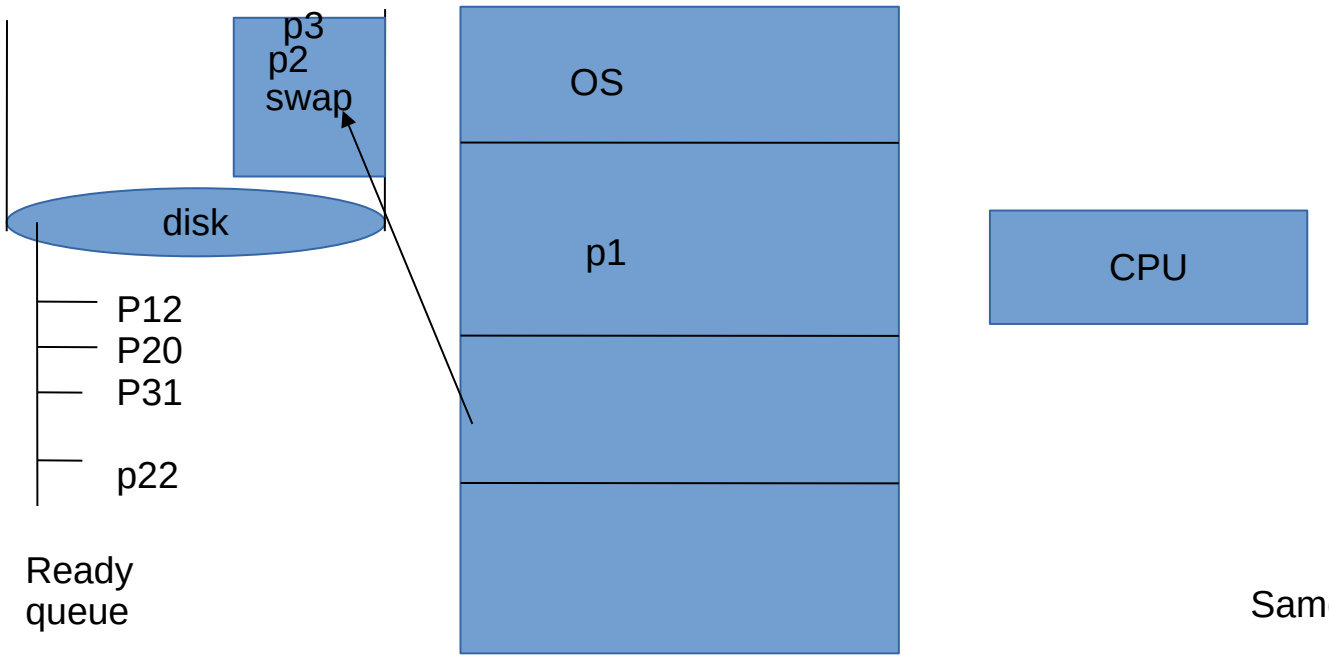

```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```

p2

When p2 is doing I/O then, is p2 required to be in main memory?

NO

Same thing might happened to p3 also



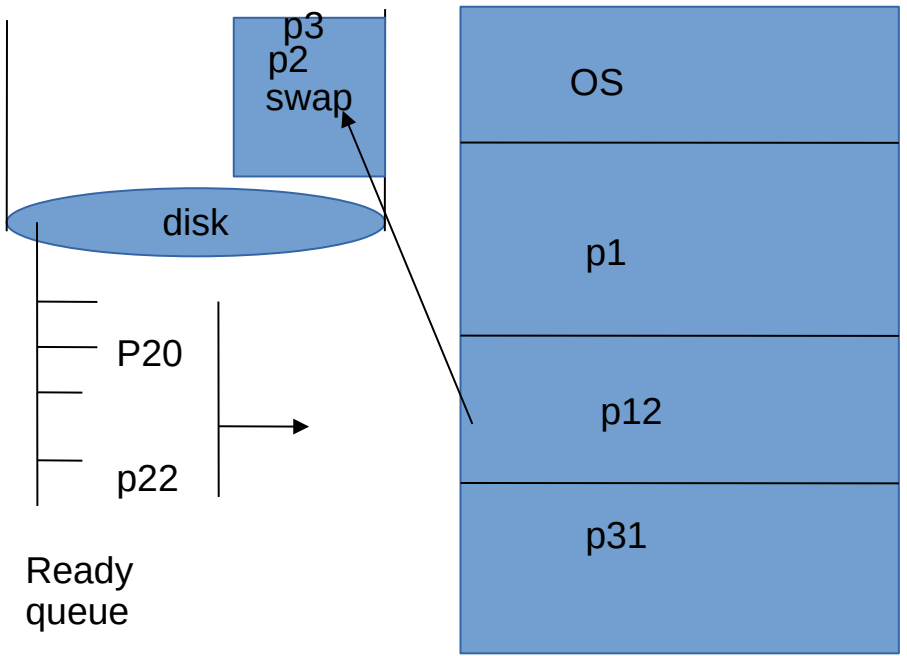

```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```

p2

When p2 is doing I/O then, is p2 required to be in main memory?

NO

Same thing might happened to p3 also



CPU

```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```

p2

When p2 is doing I/O then, is p2 required to be in main memory?

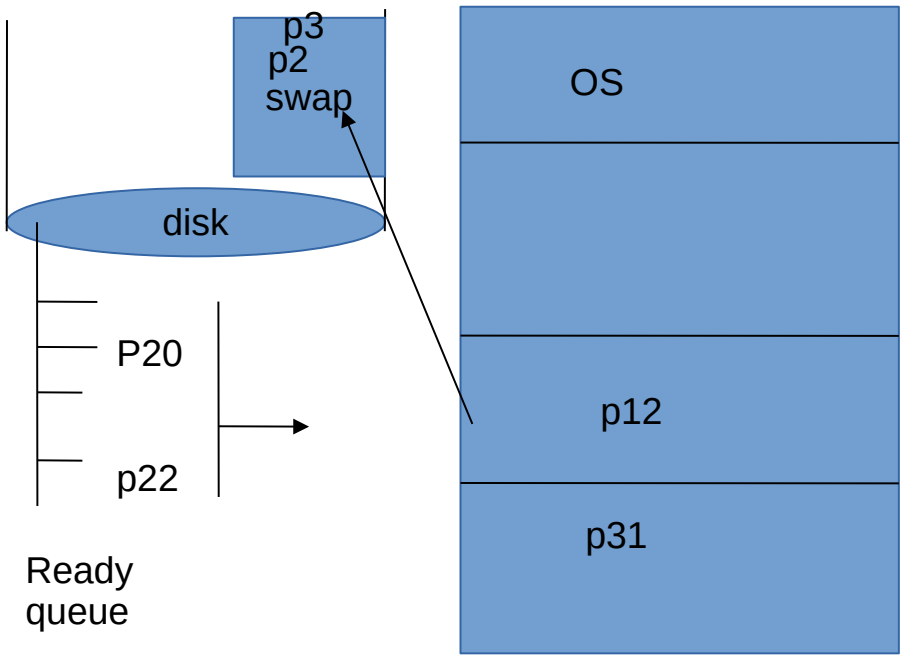
NO

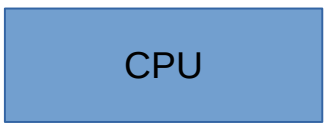
Same thing might happened to p3 also

In the free memory only one can be fitted.. who will get the chance to be in main memory?

By the time p1 finish its operation and both p2 and p3 finish I/O..

```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```





p2

When p2 is doing I/O then, is p2 required to be in main memory?

NO

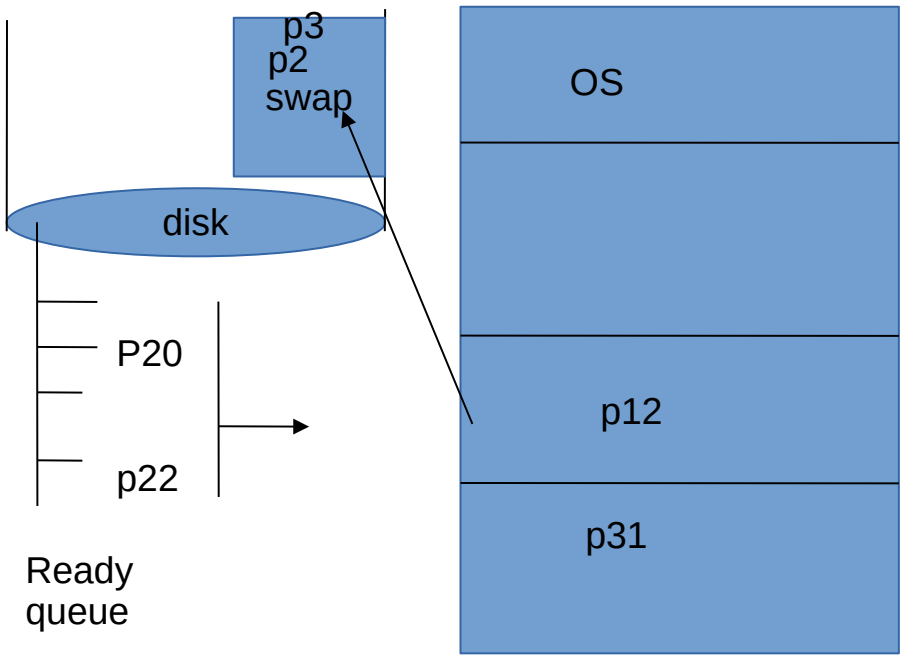
Same thing might happened to p3 also

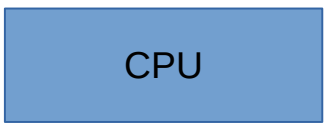
Middle term scheduler

In the free memory only one can be fitted.. who will get the chance to be in main memory?

By the time p1 finish its operation and both p2 and p3 finish I/O..

```
#include<stdio.h>
Int main()
{
        Int x=0;
        x=x+1;
        While(x<10)
        {
                printf("\n %d",x);
                x=x+1;
        }
        Return 0;
}
```





p2

When p2 is doing I/O then, is p2 required to be in main memory?

NO

Same thing might happened to p3 also

Process synchronization

A Co-operating process is one that can effect or be effected by the other co-operating processes executing in the system.

Co-operating process often share some common storage that can be read or write by both processes.

Storage may be a variable or a file ….

Concurrent access of the storage may result in data inconsistency.