Memory Management

Thursday, 9 July 2026 12:53 AM

Let's see what happens when we compile & run a C program

→
```
#include <stdio.h>
int main(){
        char str[] = "Hello";
        printf("%s", str);
}
```

gcc hello.c

Executable (a.out) → ./a.out → process

stored on hdd

execute from RAM

Process →

→ A program in execution.

→ Present in the RAM

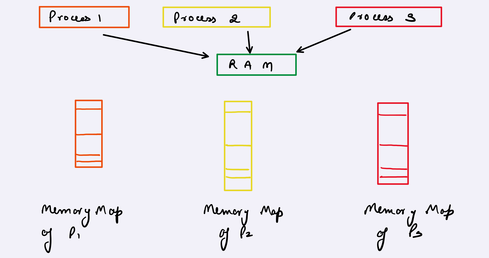
→ Comprises of
- Executable instructions
- Stack
- Heap
- State in OS (in kernel)

→ State contains: registers, list of open files, related processes etc.

Sharing of RAM

We can have multiple process running simultaneously and sharing the RAM. But RAM is a limited resource.

Now irrespective of the fact on how multiple processes are managed by OS, their memory map should always be present in the RAM

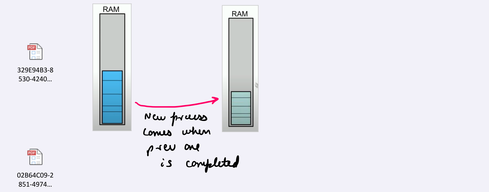


Let's see how RAM manages these processes.

★ Single Contiguous Model

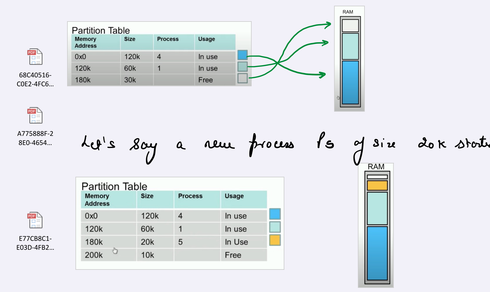
→ No sharing
- One process occupies RAM at a time
- when one process completes, another process is allocated RAM.

→ Process memory size is restricted by RAM size.

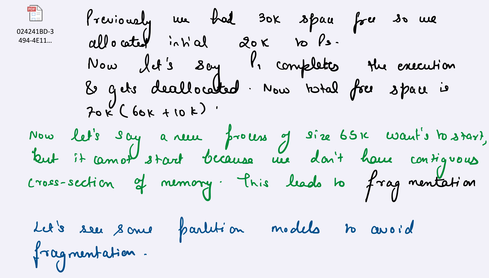


★ Partition Model

As long as sufficent contiguous space is available, new processes are allocated in the memory.



Let's say a new process P5 of size 20K started



024241BD-3
494-4E11...

Previously we had 30K space free so we allocated initial 20K to P5.
Now let's say P1 completes the execution & gets deallocated. Now total free space is 70K (60K + 10K).

Now let's say a new process of size 65K want's to start, but it cannot start because we don't have contiguous cross-section of memory. This leads to fragmentation

Let's see some partition models to avoid fragmentation.

Partition Model (finding the right fit)

1. First fit (may make fragmentation worse)
2. Best fit (may affect performance)
   → finds the most correct free area.

Note → deallocation of memory can have overhead of merging free partitions to reduce fragmentation

Modern day OS do much better in memory management using virtual memory & segmentation.