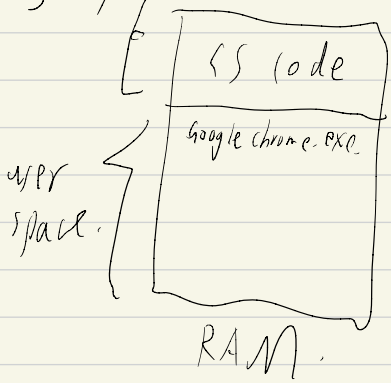
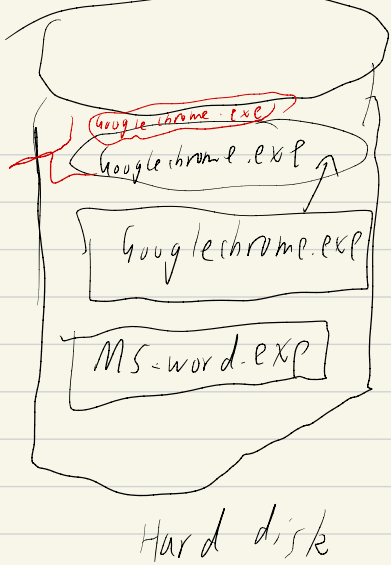
(I 37) → OS space





a copy...
a process...

e.g. 1.
for a single program, multiple processes can be created.
Google Chrome.exe

e.g. 2.
source code —(compiler)→ machine code ⟶ RAM ⟶ CPU
OR
→ stay in the RAM, waiting for some events to happen (CPU, I/O).

Difference between program and process:
A program in execution is called as process.

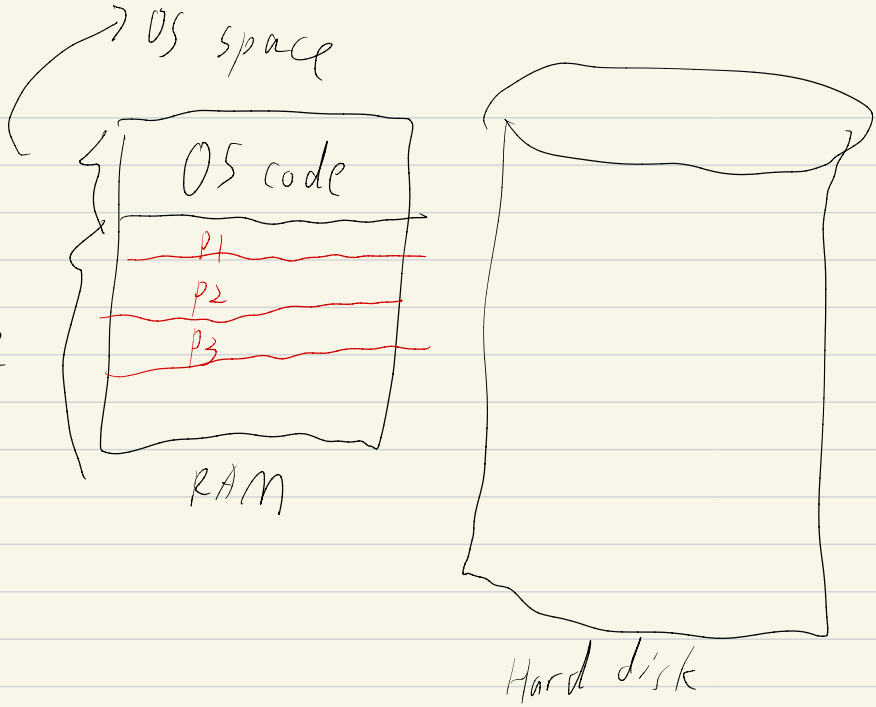
States of a process

1. New state (created)
2. Ready state (in the RAM but neither executed nor processed)
3. Running state (be processed by CPU line by line / by CPU)
4. I/O state (performing I/O event)
5. Terminated state
6. Suspend Ready state
7. Suspend Wait state

I/O state is also called block state.
Why?

Once a process has completed its execution, which means it has completed with everything, it will be completely removed from RAM.

(I ≥ 8)



Assumption { size of RAM = 4GB
size of single process = 4KB.

(*) $\frac{\text{size of RAM}}{\text{size of single process}} = \frac{2^{32}\,B}{2^{12}\,B} = 2^{20}$ processes inside my RAM.

1 MB = 2^{10} Bytes 1 MB = 2^{20} Bytes 1 GB = 2^{30} Bytes
1 TB = 2^{40} Bytes 1 Byte = 8 bits

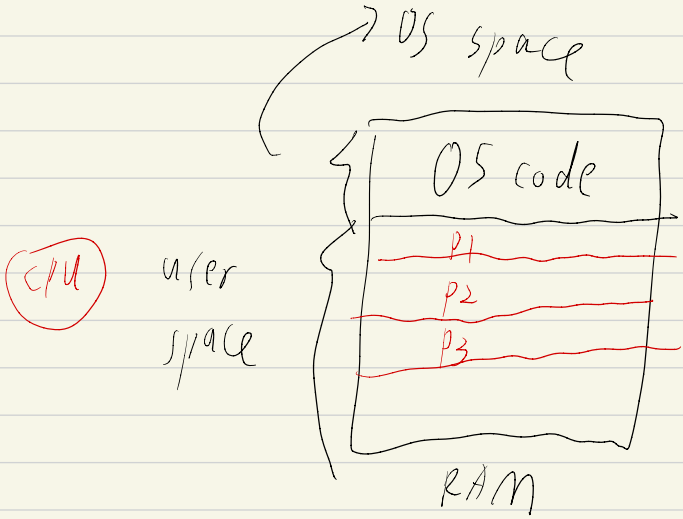
Degree of Multiprogramming:

Maximum number of processors which can be presented in the RAM of our computer.

(I 3 9), Types of OS

1) Batch
2) multiprogramming OS
3) multiprocessing OS

Batch operating system: it means the degree of multiprogramming is always one. And only one CPU.



I/O devices to work in parallel.
... I'll be placing one program in the RAM.

After completing of this process I will be fetching the next process from the Hard disk.

See, execution and I/O cannot be done at the same time for a particular process. It can be done for two processes... which means when a process is being executed by CPU we can actually perform I/O for another process that can be done.

CPU efficiency

$$= \frac{\text{Useful time of CPU}}{\text{Total time of CPU}}$$

//

(multiprocess could enhance CPU efficiency.

one process undergoes I.O.

another ~ is being processed by CPU.)

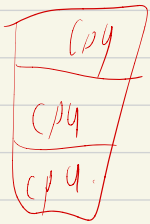
//

Multiprogramming OS system

↳ But the CPU is still only ONE.

Multiprocessing OS system:

could have more than one CPU.



↳ CPU efficiency will be enhanced.

Difference between parallel processing VS concurrent processing

e.g.1 mom use 2 spoons to feed 2 children using two hands.
(parallel processing)

e.g.2. one hand to feed one kid with 1 spoon, to feed another kid with another spoon.
(concurrent processing)

parallel processing: (i) faster
(ii) in parallel processing the system cost will increase. (costlier than multiprocessing)

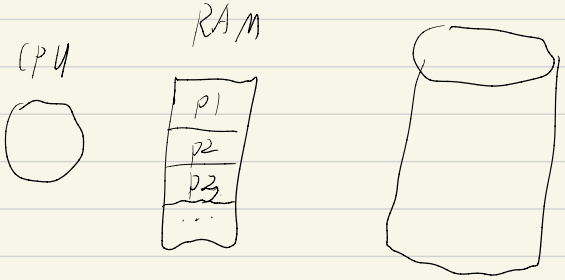
so, still multiprocessing: today:

quadroprocessor: four processors;
octa processor: : eight processors.

(I 3 10)

throughout the course only one O.S.

multiprogramming O.S. Hard Disk.



(multiprogramming O.S. in this course.)

(I 3 11)

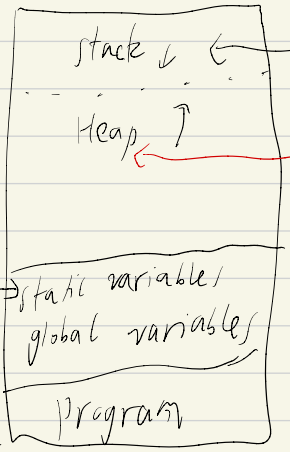
A process will not only contain program but it will have stack, heap, data post.

For every program, a process will be created whenever we open the program.

passive entity
active entity

(process control block)
PCB of a process

e.g.
static int()

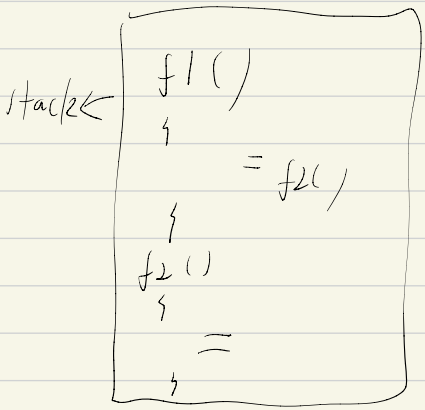


call function (dynamic)

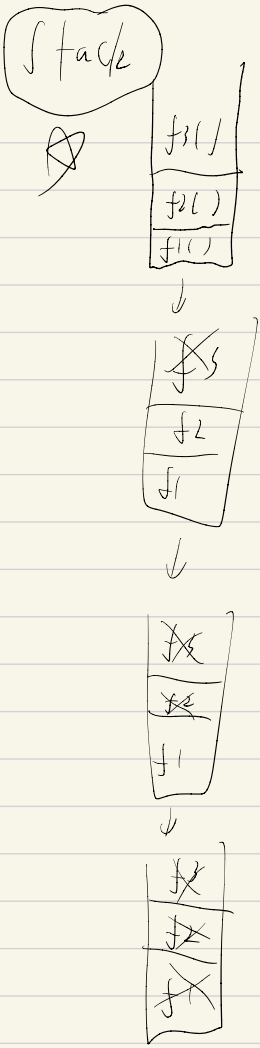
dynamic memory (at run time)

(Also called as Data Past)

(Also called as code past)



some programs may need more space for stack, some may need more for heap.



f1() call f2()

f2() call f3()

f3() executed then pop

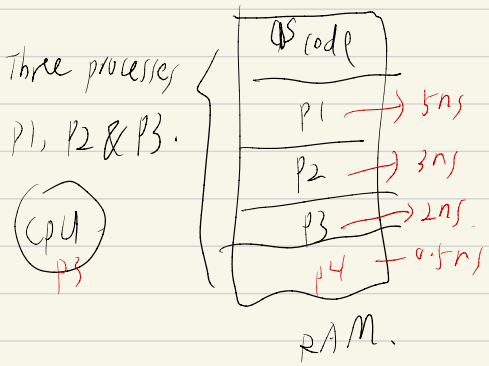
f2() . . . . . .

f1() . . . . . .

Dynamic memory allocation (allocating memory at the run time)

- malloc( )
- calloc( )

see sec3_11_1.c.



Operating System code (OS code):
It will be used in order to define which among the 3 processes should be selected by the CPU.
(i.e. Scheduler)

Scheduler is nothing but a program which is a part of our OS program. It has the code to say that... in case if more than one process needs the CPU... which among them should be executing the CPU first...)

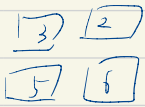
it will pick P1 or P2 or P3 to be picked by the CPU first.

e.g.
(i) FIFO scheduling:
(ii) shortest job first algorithm will be executed first. If P4 appears, then P3 will be forcefully stopped. Then P4 will be

completed. we have to start again P3 from the place we stopped.

Process Attributes.

1) process id (a unique No. for every process)
2) program counter (e.g. Line 4)
   (is a register which is maintained by CPU)
3) Process state
4) General purpose registers
5) Priority
6) List of open files
7) List of open devices
8) Protection.



For every process we will be having a set of register values. And these register values have to be remembered wherever we are preempted from process from execution by the CPU.

Why? Due to scheduling algorithms { attribute (4) ~ (6) ~ (7) e.g: printer

A particular process has stack space and heap space, which means, for function calls it uses the stack space and for dynamic allocation it uses the heap space.

protection { heap space, stack space

A process should not access another process's space of protection.