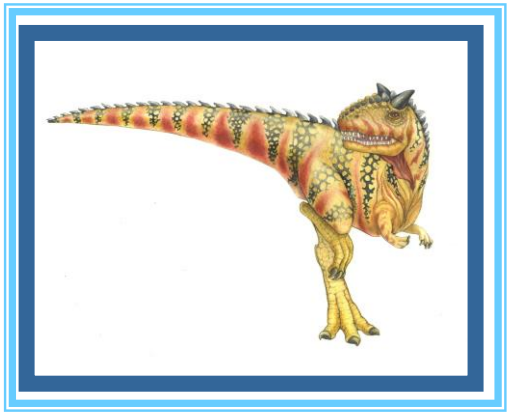
Processes
Day2: March 2022

Kiran Waghmare



Who can see what you share here? Re

Day 2: 15-03-2022

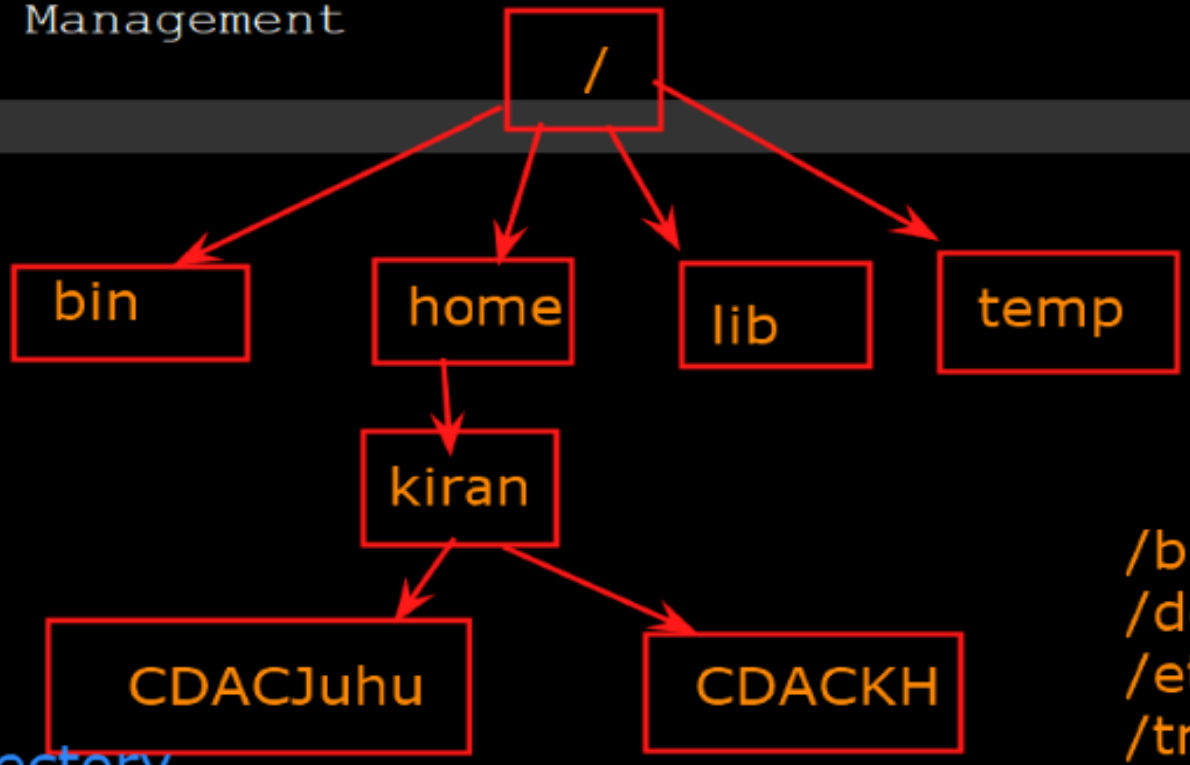
==========

Topics:

-Linux Commands

-Process Management

File system:



/bin
/dev
/etc
/tmp

. => directory

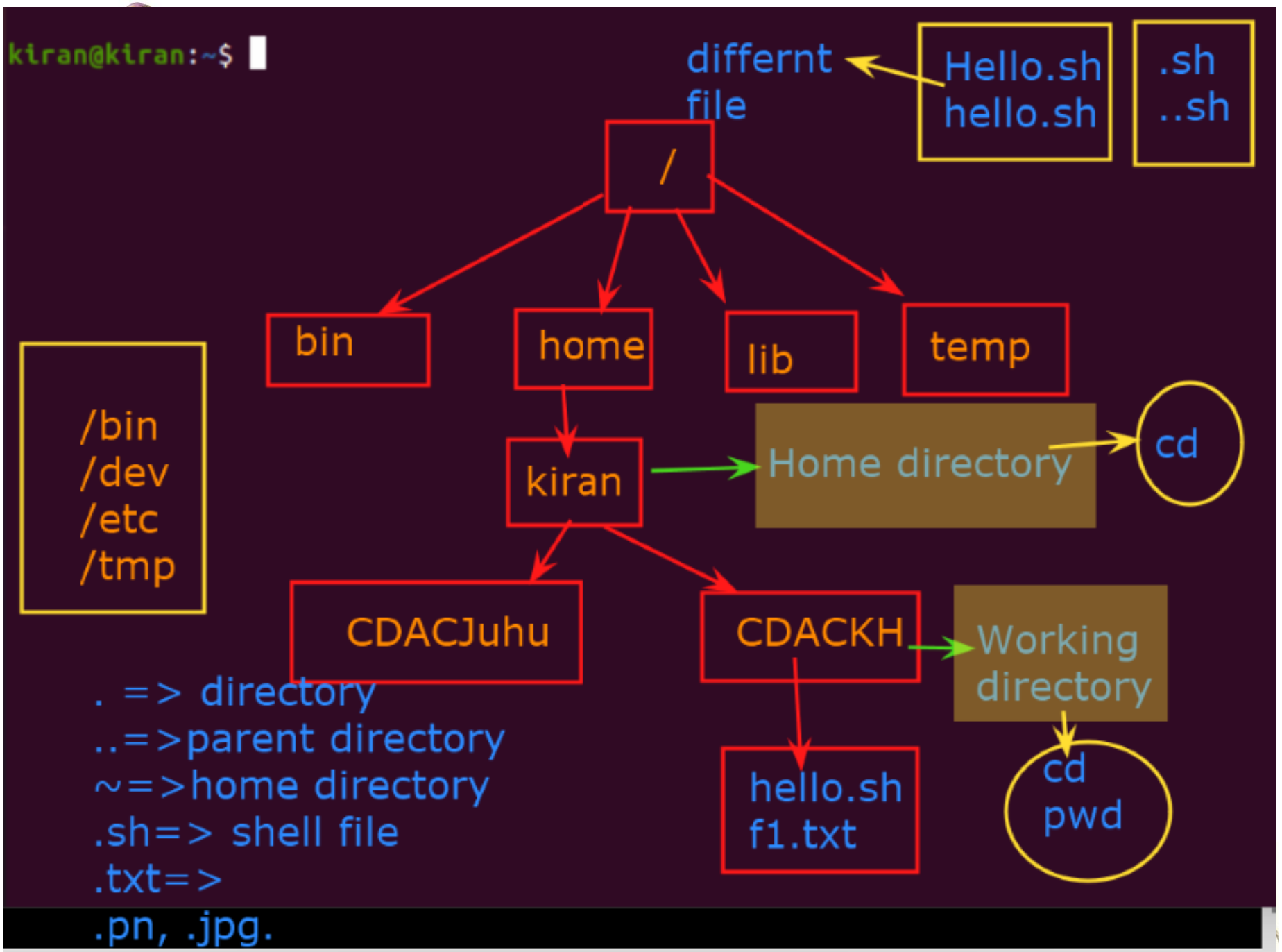
..=>parent directory

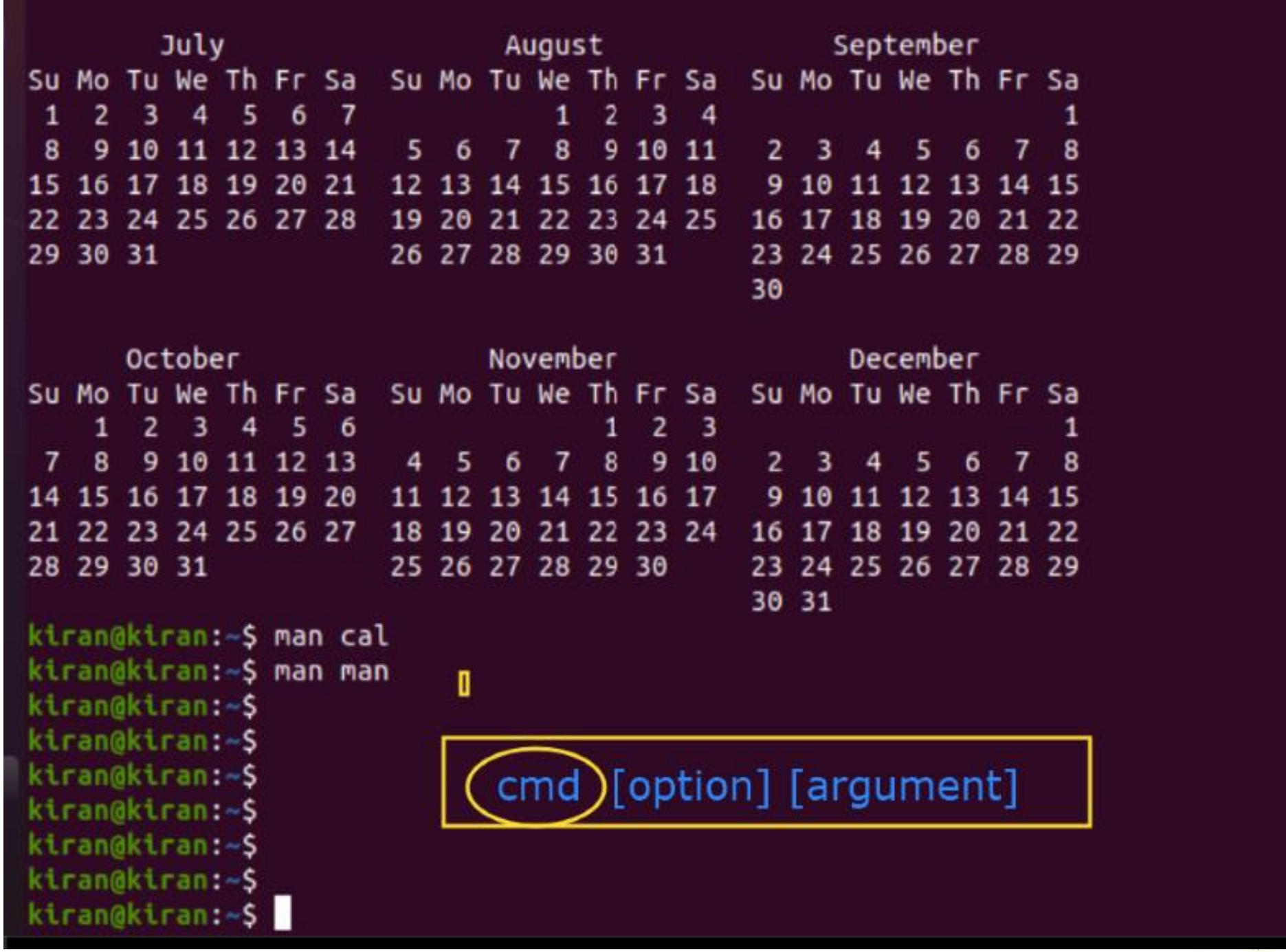
~=>home directory

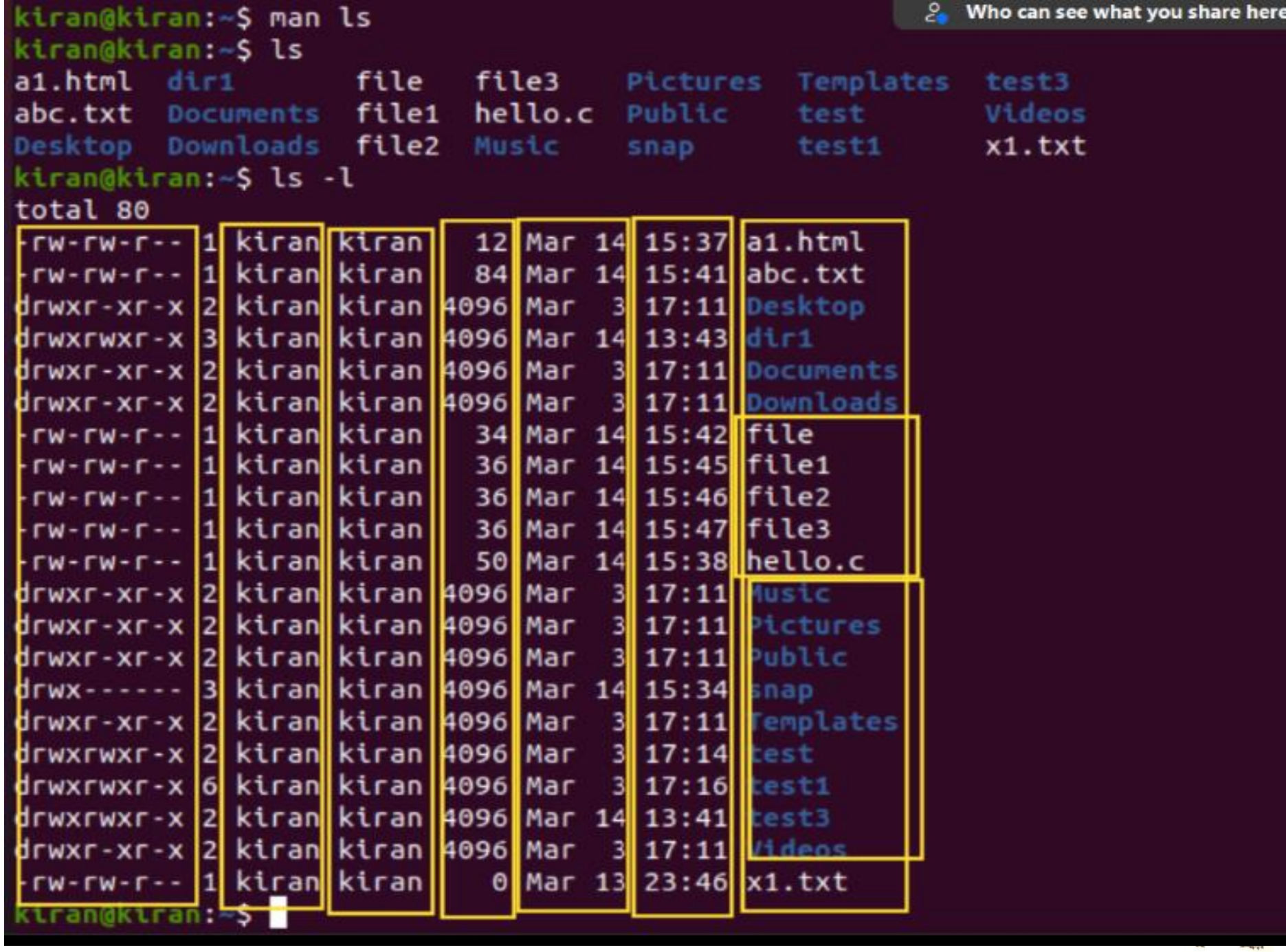
.sh=> shell file

.txt=>

.pn, .jpg.


kiran@kiran:~$
differnt file
Hello.sh
hello.sh
.sh
..sh
/
bin
home
lib
temp
/bin
/dev
/etc
/tmp
kiran
Home directory
cd
CDACJuhu
CDACKH
Working directory
. => directory
..=>parent directory
~=>home directory
.sh=> shell file
.txt=>
hello.sh
f1.txt
cd
pwd
.pn, .jpg.




kiran@kiran:~$ man ls
kiran@kiran:~$ ls
a1.html dir1 file file3 Pictures Templates test3
abc.txt Documents file1 hello.c Public test Videos
Desktop Downloads file2 Music snap test1 x1.txt
kiran@kiran:~$ ls -l
total 80
-rw-rw-r-- 1 kiran kiran 12 Mar 14 15:37 a1.html
-rw-rw-r-- 1 kiran kiran 84 Mar 14 15:41 abc.txt
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Desktop
drwxrwxr-x 3 kiran kiran 4096 Mar 14 13:43 dir1
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Documents
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Downloads
-rw-rw-r-- 1 kiran kiran 34 Mar 14 15:42 file
-rw-rw-r-- 1 kiran kiran 36 Mar 14 15:45 file1
-rw-rw-r-- 1 kiran kiran 36 Mar 14 15:46 file2
-rw-rw-r-- 1 kiran kiran 36 Mar 14 15:47 file3
-rw-rw-r-- 1 kiran kiran 50 Mar 14 15:38 hello.c
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Music
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Pictures
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Public
drwx------ 3 kiran kiran 4096 Mar 14 15:34 snap
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Templates
drwxrwxr-x 2 kiran kiran 4096 Mar 3 17:14 test
drwxrwxr-x 6 kiran kiran 4096 Mar 3 17:16 test1
drwxrwxr-x 2 kiran kiran 4096 Mar 14 13:41 test3
drwxr-xr-x 2 kiran kiran 4096 Mar 3 17:11 Videos
-rw-rw-r-- 1 kiran kiran 0 Mar 13 23:46 x1.txt
kiran@kiran:~$
Who can see what you share here


kiran@kiran:~/cdac1/Mumbai$ cat > aa1.txt
Pratik
Amol
Amit
Aditya
Yogesh
sara
Siddhesh
Shivanjali
Sonal
Sonm
Himanshu
kiran@kiran:~/cdac1/Mumbai$ ls
aa1.txt
kiran@kiran:~/cdac1/Mumbai$ cat aa1
cat: aa1: No such file or directory
kiran@kiran:~/cdac1/Mumbai$ cat aa1.txt
Pratik
Amol
Amit
Aditya
Yogesh
sara
Siddhesh
Shivanjali
Sonal
Sonm
Himanshu
kiran@kiran:~/cdac1/Mumbai$
Insert File content
Read File content

```
Himanshu
Pratik
Rohit
Sakshi
sara
Shivanjali
Siddhesh
Sonal
Sonm
Summit
Yogesh
kiran@kiran:~/cdac1/Mumbai$ sort -r aa1.txt
Yogesh
Summit
Sonm
Sonal
Siddhesh
Shivanjali
sara
Sakshi
Rohit
Pratik
Himanshu
Dhiraj
Anurag
Amol
Amit
Aditya
kiran@kiran:~/cdac1/Mumbai$
```

Who can see what you share here? Rec

UNIX Filter

$head	$tail	$h1
$cut	$paste	$sort
$tr	$tree	$red
$grep	$tgrep	$egrep

Vi Editor:

1. vi filename.sh
2. Esc + I (insert)
3. Add your code in file
4. Esc + :wq

   w: save your file
   q: exit from file

5. chmod +x filename.sh

   Premission grant

6. Execute: ./filename.sh

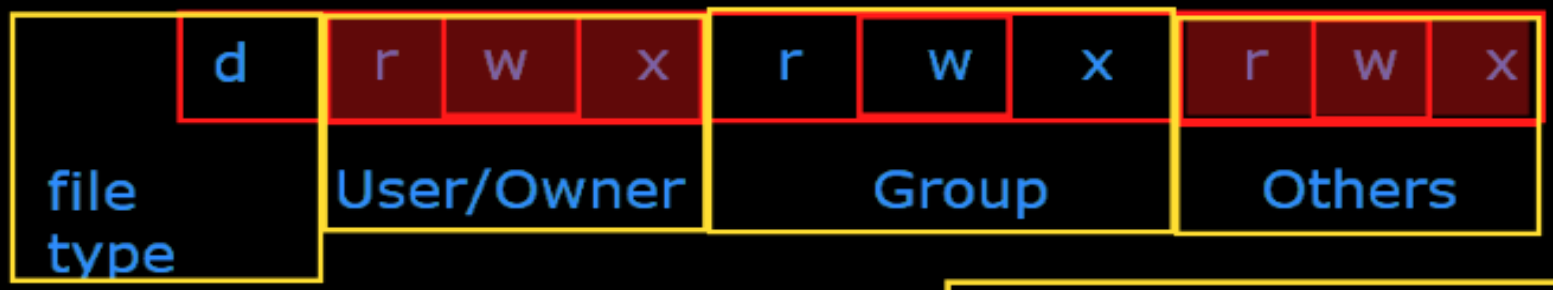
or

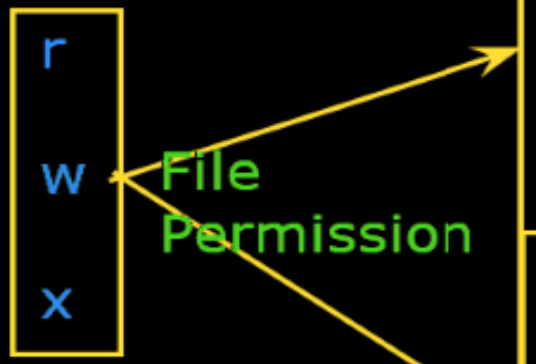
Execute: bash filename.sh

Editor:

- QED
- Vi
- Vim
- Gedit
- Nano
- Edit+

File Permissions

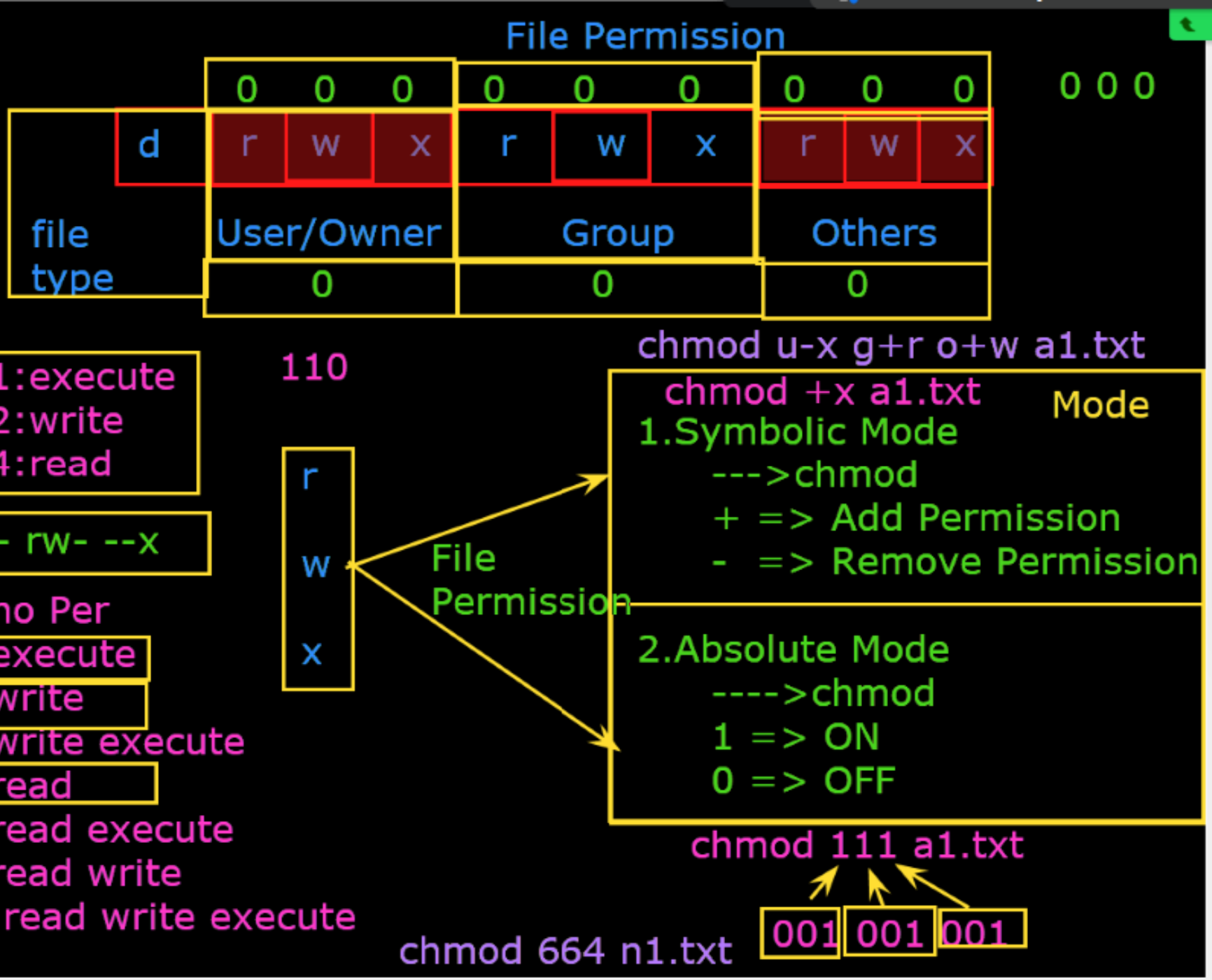




Mode

1. Symbolic Mode
   - --->chmod
   - + => Add Permission
   - - => Remove Permission

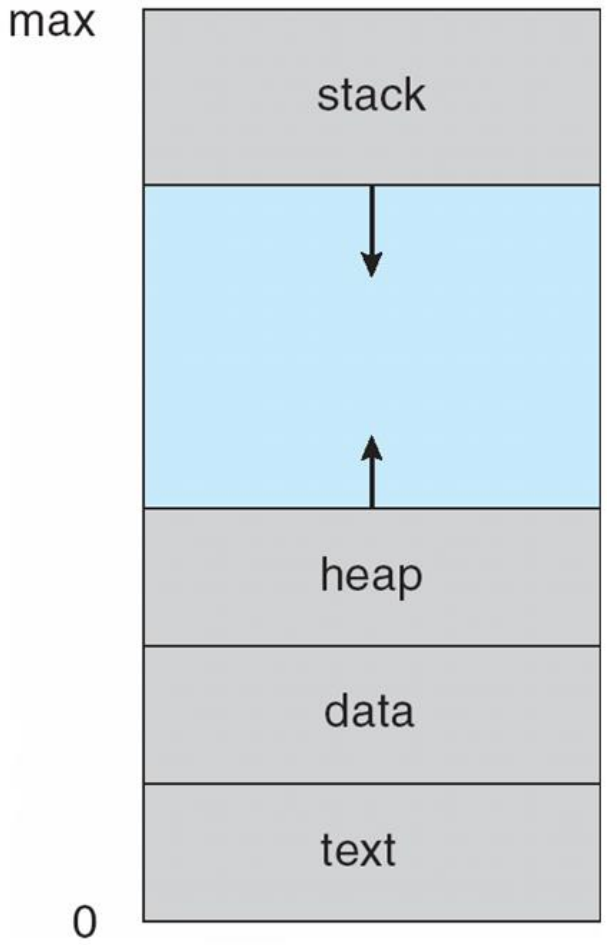
2. Absolute Mode
   - ---->chmod
   - 1 => ON
   - 0 => OFF


File Permission
0 0 0 0 0 0 0 0 0
0 0 0
d r w x r w x r w x
file type
User/Owner
Group
Others
0 0 0
chmod u-x g+r o+w a1.txt
1:execute
2:write
4:read
110
rw- rw- --x
r
w
x
File Permission
chmod +x a1.txt
Mode
1.Symbolic Mode
--->chmod
+ => Add Permission
- => Remove Permission
2.Absolute Mode
---->chmod
1 => ON
0 => OFF
0:no Per
1:execute
2:write
3:write execute
4:read
5:read execute
6:read write
7: read write execute
chmod 111 a1.txt
001 001 001
chmod 664 n1.txt

Agenda: Processes

- Preemptive and non preemptive
- Process mgmt
- Process life cycle
- Schedulers
- Scheduling algorithms
- Creation of fork, waitpid, exec system calls
- Orphan and zombie

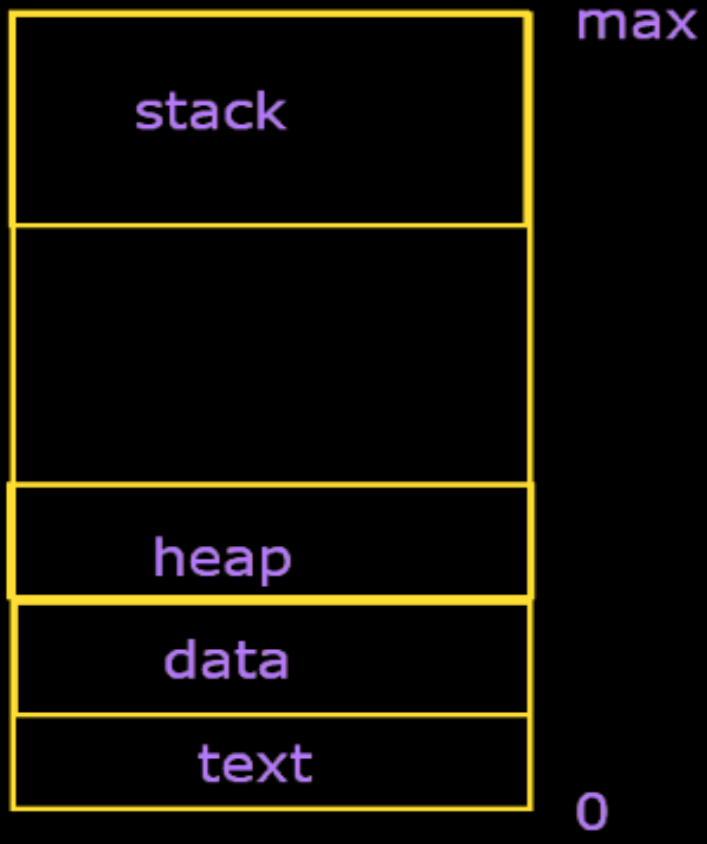
Process in Memory



Process:
-instance of computer program that being executed.
-shorter lifetime.
-dynamic instance of code and data.
-RAM memory for storage
-required resources memory, cpu, Io devices,...

Process Memory

Process: a program in execution
-sequenctial fashion for execution of program
-Program include:
    -program counter
    -stack
    -data section



Process in Operating System

- A process is a **program in execution** which then forms the basis of all computation.
- The process is not as same as program code but a lot more than it.
- A process is an 'active' entity as opposed to the program which is considered to be a 'passive' entity.
- Attributes held by the process include hardware state, memory, CPU, etc.

- **Process memory** is divided into four sections for efficient working :
- The **Text section** is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- The **Data section** is made up of the global and static variables, allocated and initialized prior to executing the main.
- The **Heap** is used for the dynamic memory allocation and is managed via calls to new, delete, malloc, free, etc.
- The **Stack** is used for local variables. Space on the stack is reserved for local variables when they are declared.

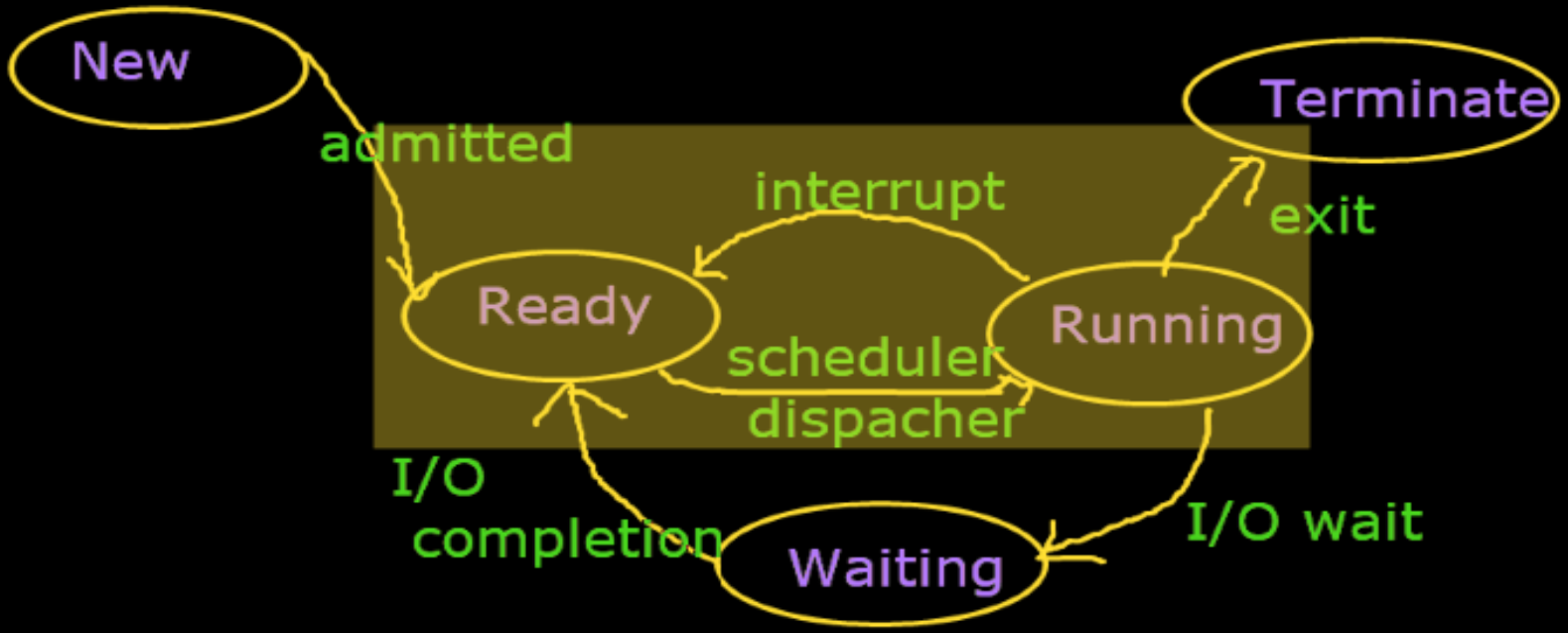
Process Concept

- An operating system executes a variety of programs:
  - Batch system – jobs
  - Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably
- **Process – a program in execution; process execution must progress in sequential fashion**
- A process includes:
  - program counter
  - stack
  - data section

-finished execution

Process Control Block:

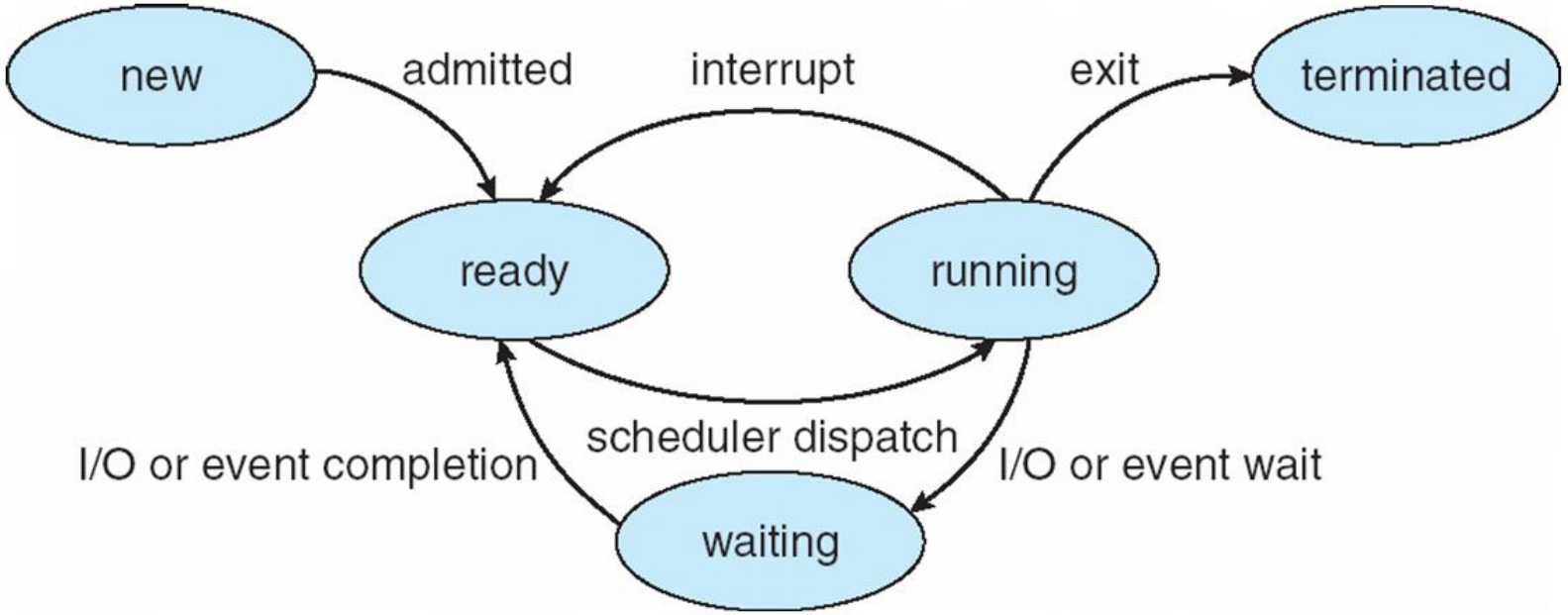
-Process state
-Program counter
-CPU registers
-CPU Scheduling informotion
-Memory Management information
- Accounting information
-I/O status information



Process State

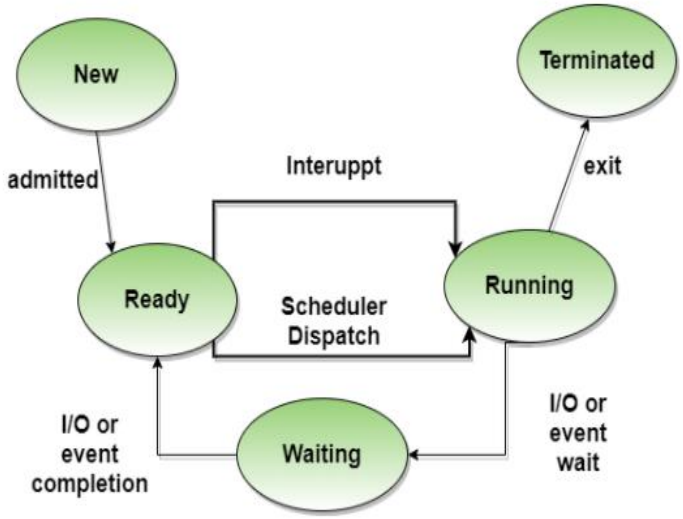
- As a process executes, it changes *state*
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Diagram of Process State



The different Process States

- Processes in the operating system can be in any of the following states:
- NEW- The process is being created.
- READY- The process is waiting to be assigned to a processor.
- RUNNING- Instructions are being executed.
- WAITING- The process is waiting for some event to occur(such as an I/O completion or reception of a signal).
- TERMINATED- The process has finishe

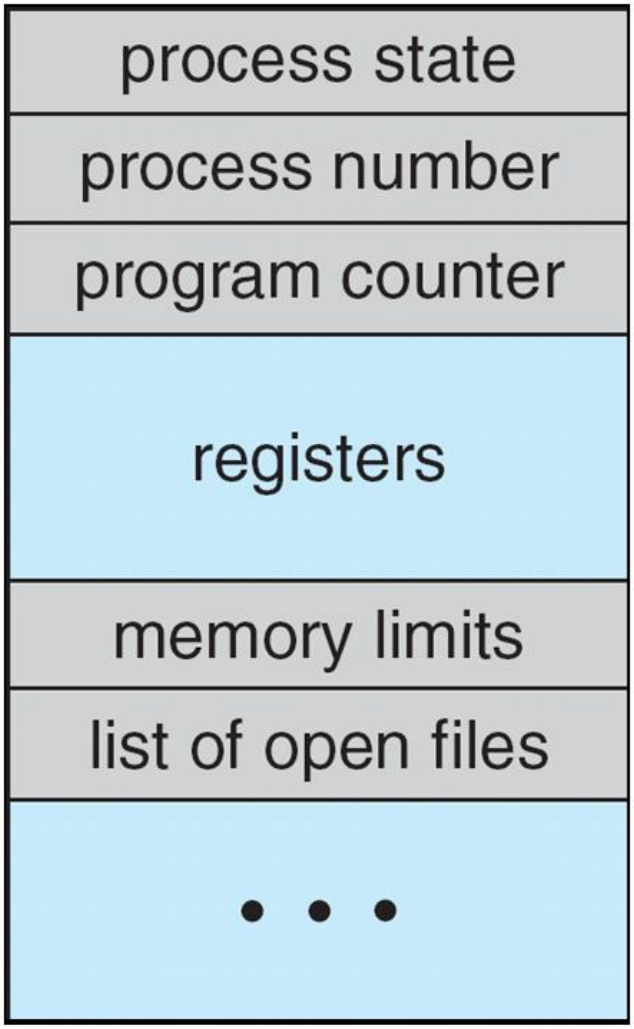


Process Control Block (PCB)

Information associated with each process

- Process state
- Program counter
- CPU registers
- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information

Process Control Block (PCB)



What is Process Scheduling?

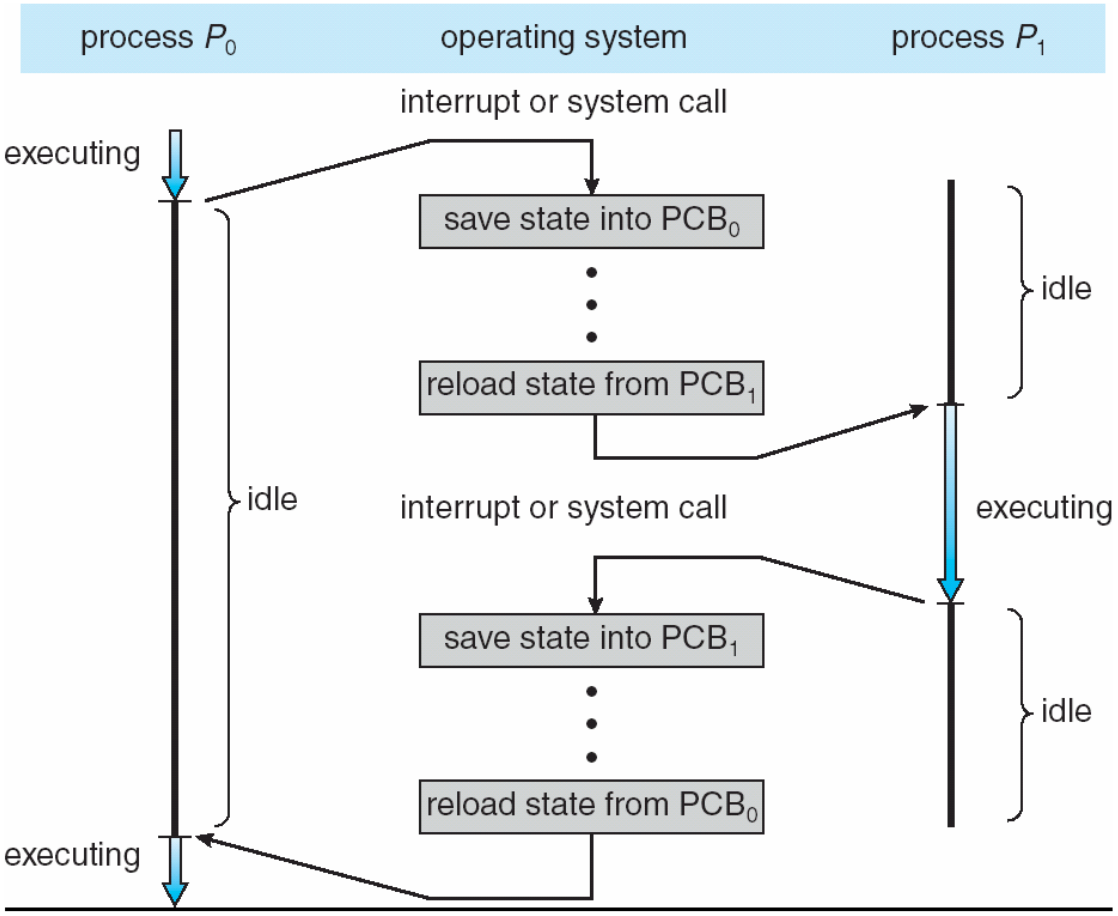
- The act of determining which process is in the ready state, and should be moved to the running state is known as **Process Scheduling**.

- The prime aim of the process scheduling system is to **keep the CPU busy all the time and to deliver minimum response time for all programs**. For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU.

Scheduling fell into one of the two general categories:

- **Non Pre-emptive Scheduling:** When the currently executing process gives up the CPU voluntarily.
- **Pre-emptive Scheduling:** When the operating system decides to favour another process, pre-empting the currently executing process.

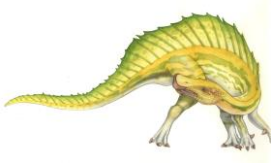
CPU Switch From Process to Process

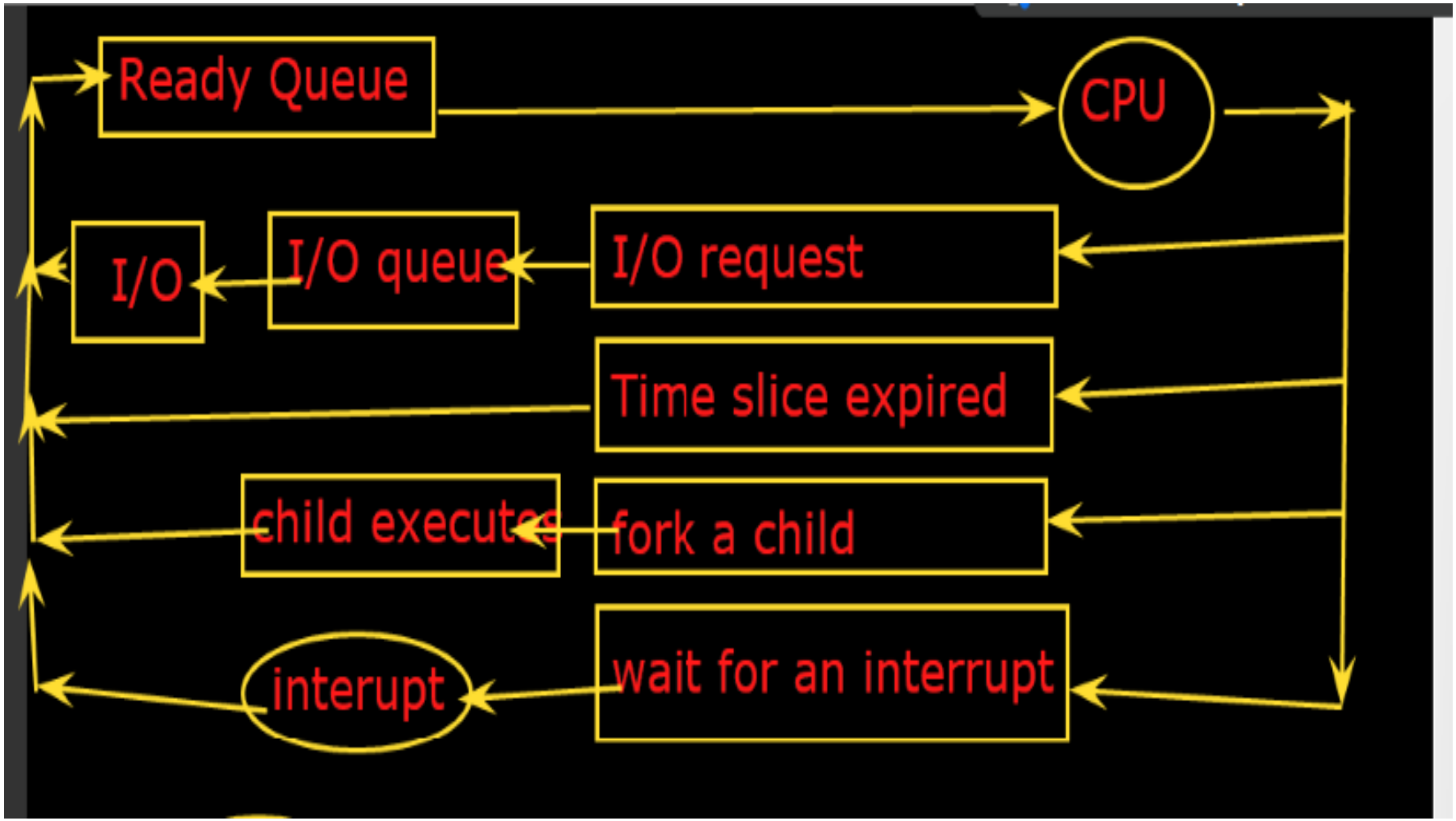


Process Scheduling

- When there are two or more runnable processes then it is decided by the Operating system which one to run first then it is referred to as Process Scheduling.
- A scheduler is used to make decisions by using some scheduling algorithm.
- Given below are the properties of a **Good Scheduling Algorithm**:

- Response time should be **minimum** for the users.
- The **number of jobs processed per hour should be maximum** i.e Good scheduling algorithm should give maximum throughput.
- The utilization of the CPU should be 100%.
- Each process should get a fair share of the CPU.

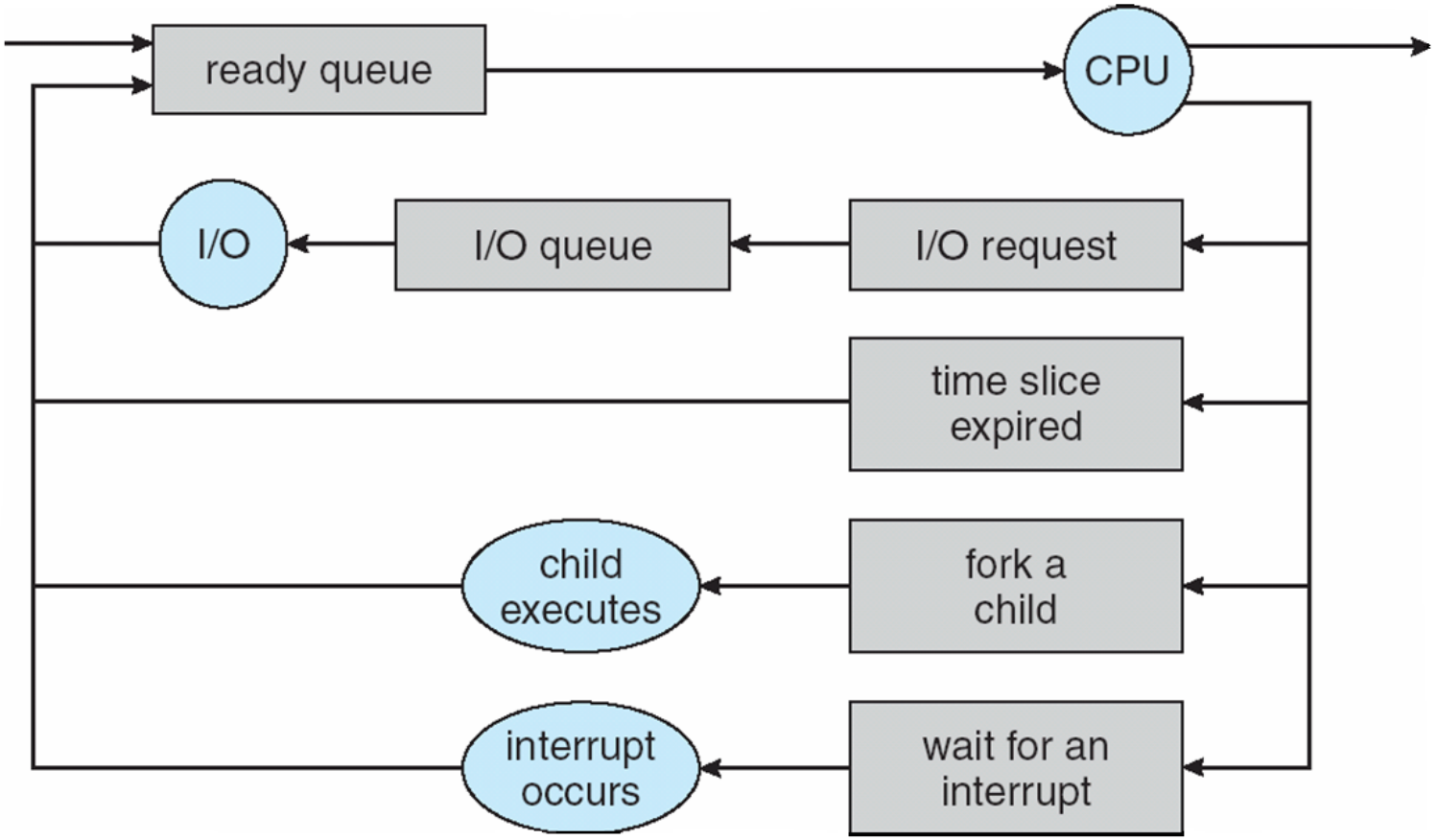



Ready Queue
CPU
I/O
I/O queue
I/O request
Time slice expired
child executes
fork a child
interupt
wait for an interrupt

Process Scheduling Queues

- **Job queue** – set of all processes in the system
- **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
- **Device queues** – set of processes waiting for an I/O device
- Processes migrate among the various queues

Representation of Process Scheduling

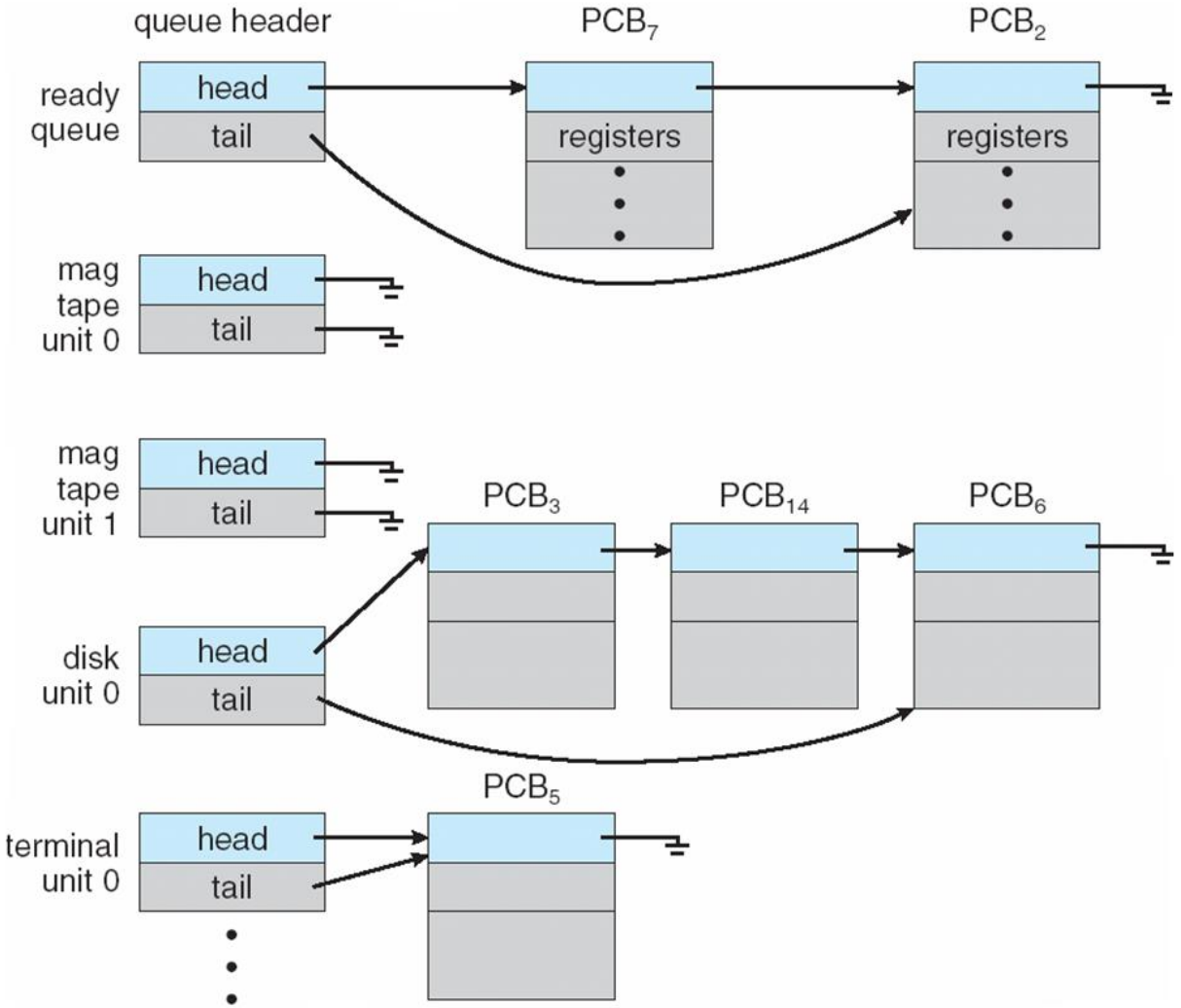


What are Scheduling Queues?

- All processes, upon entering into the system, are stored in the **Job Queue**.
- Processes in the Ready state are placed in the **Ready Queue**.
- Processes waiting for a device to become available are placed in **Device Queues.** There are unique device queues available for each I/O device.
- A new process is initially put in the Ready queue. It waits in the ready queue until it is selected for execution(or dispatched). Once the process is assigned to the CPU and is executing, one of the following several events can occur:

- The process could issue an I/O request, and then be placed in the I/O queue.
- The process could create a new subprocess and wait for its termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.

Ready Queue And Various I/O Device Queues



Types of Schedulers

- There are three types of schedulers available:

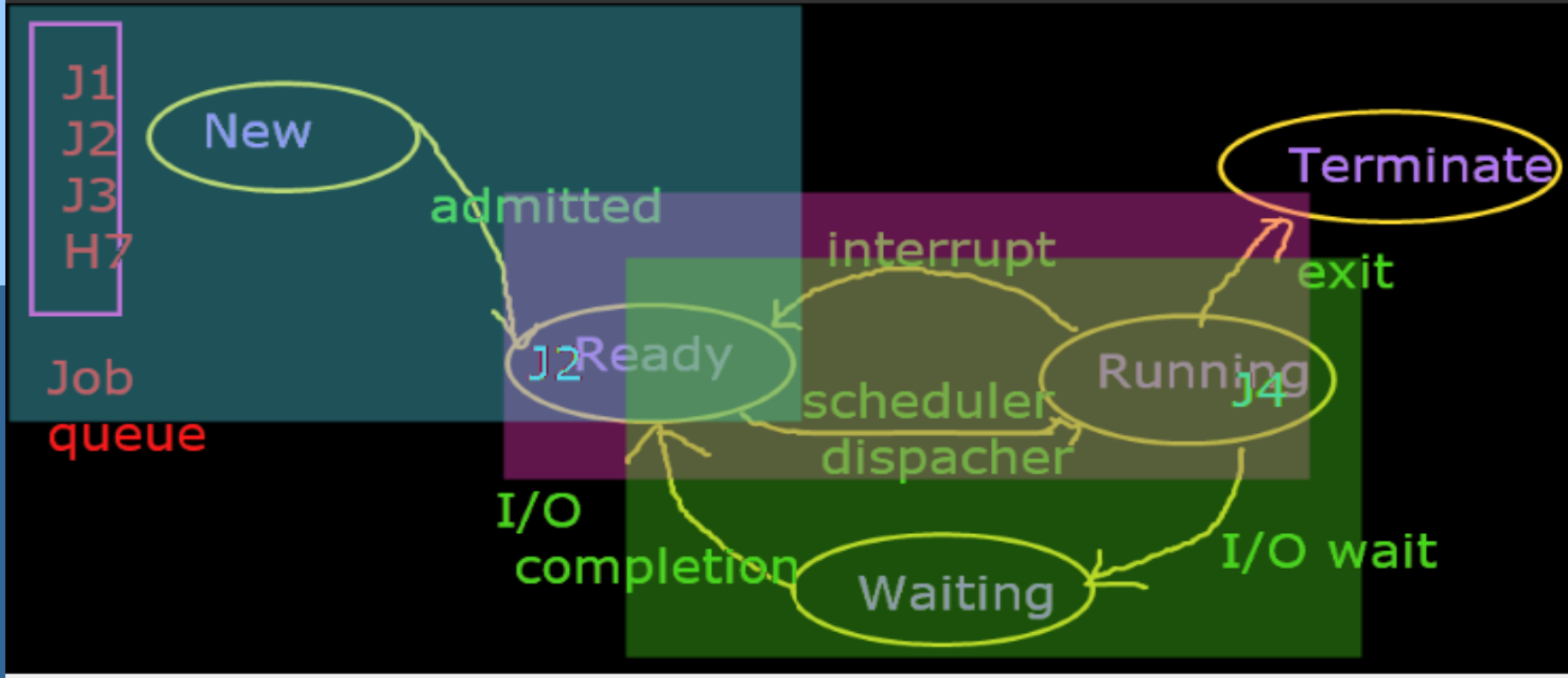
- Long Term Scheduler
- Short Term Scheduler
- Medium Term Scheduler

1.Long Term Scheduler (Job Scheduler)
-selects which process should be brought into the ready queue

2.Short Term Scheduler (CPU Scheduler)
-selects which process should be execute next and allocates CPU

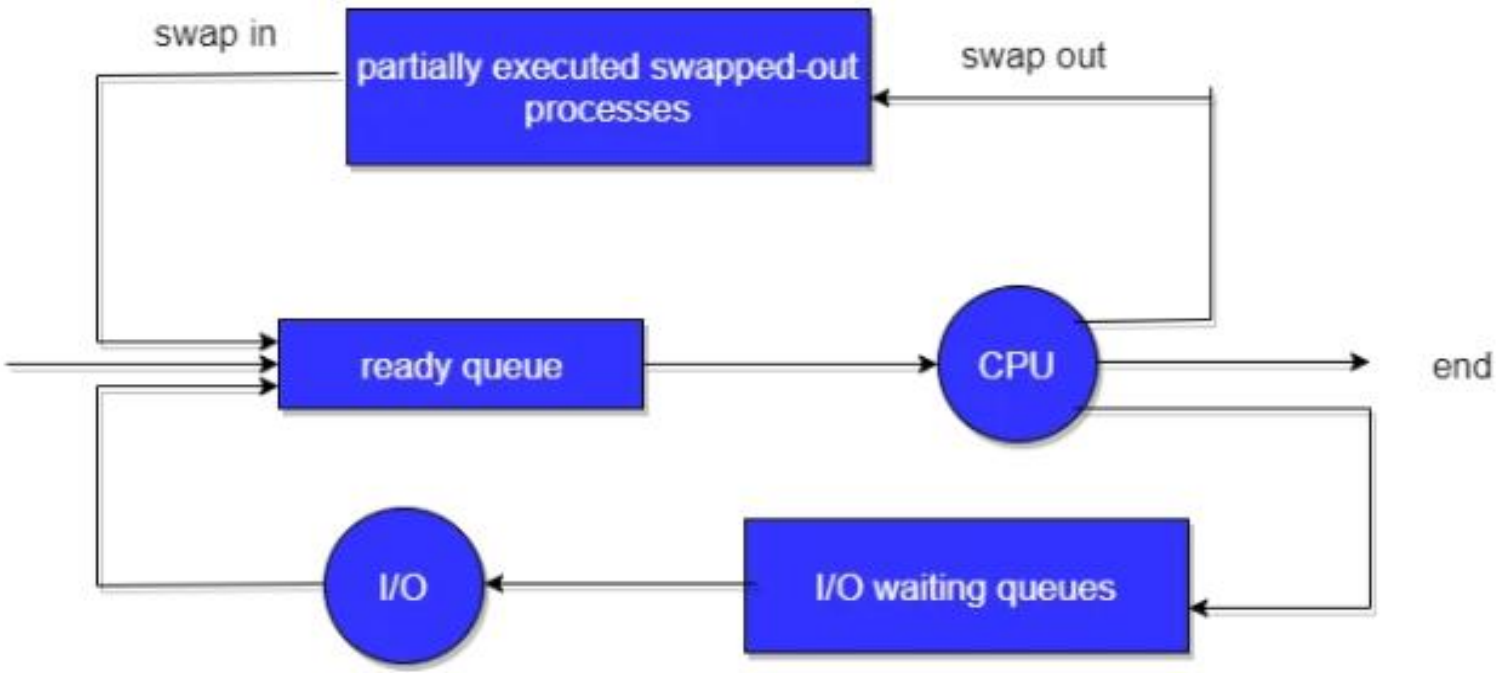
3.Midium Term Scheduler
-process swapping (swap in and swap out)

Good Scheduling Algorithm:



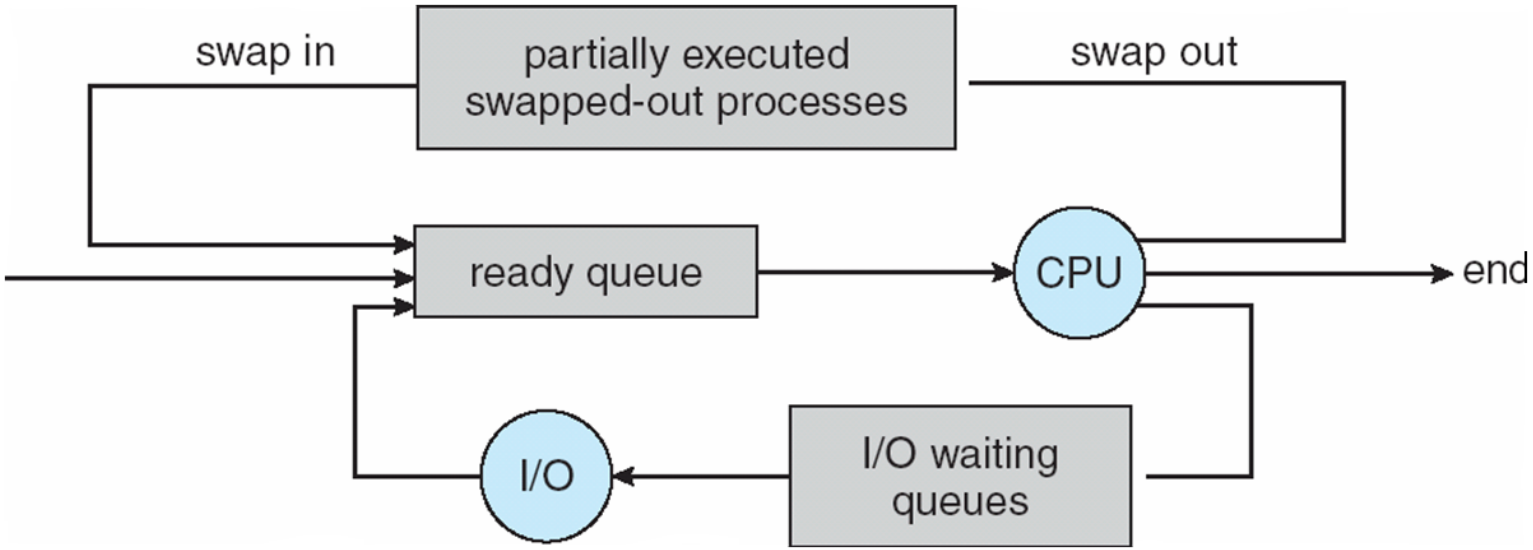
Schedulers

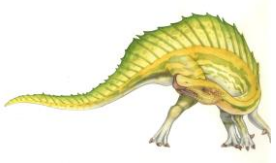
- **Long-term scheduler** (or job scheduler) – selects which processes should be brought into the ready queue
- **Short-term scheduler** (or CPU scheduler) – selects which process should be executed next and allocates CPU



Addition of Medium-term scheduling to the queueing diagram.

Addition of Medium Term Scheduling





Schedulers (Cont)

- Short-term scheduler is invoked very frequently (milliseconds) ⇒ (must be fast)
- Long-term scheduler is invoked very infrequently (seconds, minutes) ⇒ (may be slow)
- The long-term scheduler controls the *degree of multiprogramming*
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts