Process and process management – Part 2

Agenda

1) Concurrent and Sequential execution of processes.

2) Process id and its Parent process id.

3) Some definitions

    a)Response time.

    b)Starvation time.

    c)Turnaround time.

    d) Throughput.

4) States of process.

Concurrent and sequential execution

- Before completion of one process execution if another process starts the execution then that situation is called concurrent execution.
- After the completion of one process if another process starts execution then that situation is called sequential execution.
- In multiprogramming environment concurrent execution happens.

Process id and parent process id

- Each process has a process ID(PID).It is a positive integer that uniquely identifies the process on the system.
- If we want to print pid below function we need to use.

```
#include<unistd.h>
pid_t getpid(void);
* Always successfully returns process ID of caller */
```

- The Linux kernel limits process IDs to being less than or equal to 32,767 defined by PID_MAX.
- In Linux the PID_MAX is available in the file /proc/sys/kernel/pid_max.

Process id and parent process id

- Each process has a parent—the process that created it. A process can find out the process ID of its parent using the getppid() system call.

```
pid_t getppid(void);
* Always successfully returns process ID of parent of caller *
```

Terms Related to Process

- ***Response Time***: The time gap between the process loaded into the RAM and its first instruction executed by the CPU.
- ***Starvation Time***: A process in its life cycle how much starved without executing by CPU, this time is starvation time.
- ***Turnaround Time***: time is a time gap between process loaded into the RAM and its execution completed.
- ***Throughput:*** The number of processes that are completed per time unit, called throughput.

States of process

- The process in its life time always may not executed by the CPU in multiprogramming environment.
- If the process is not executing by the CPU Then the process may present in the below states.

  1) Ready state

  2) Wait state

  3) Delay state

  4) Suspend state.
- If process executing by CPU then we need to say it is in Running state.

States of Process (Contd..)

1)When you execute the program as ./a.out, process is created. It goes into ready state.

2)From ready state it goes to running state, CPU starts executing it and you get Hi printed on screen.

```
#include<stdio.h>
main()
{
int i;
 printf("Hi...\n");
scanf("%d",&i);
printf("i=%d\n",i);
}
```

States of Process (Contd..)

3)Then scanf() function need input from keyboard, so process enters I/O wait state.

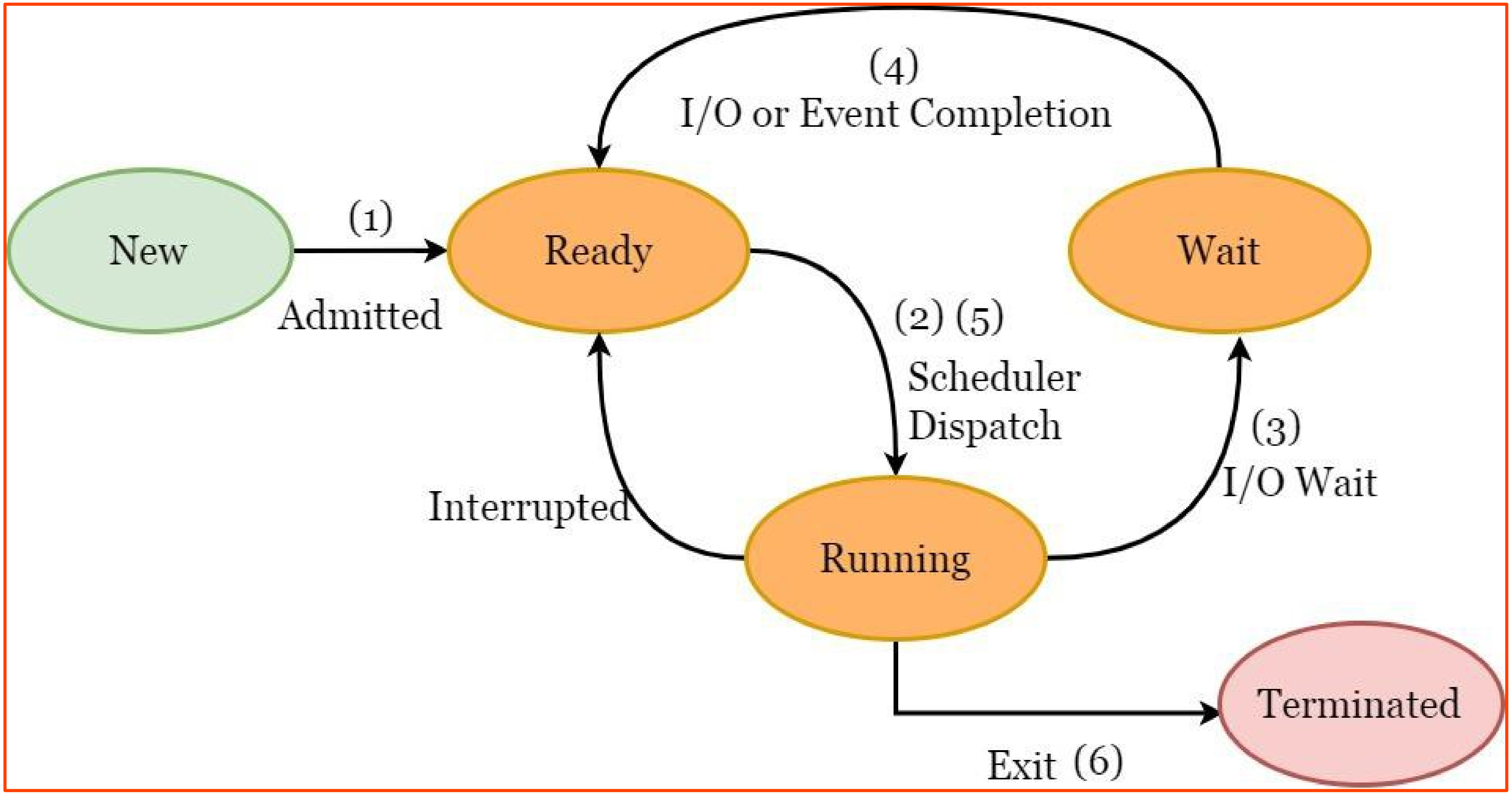
4)When user enters value, I/O request completes and process goes again to ready state. Meanwhile CPU is engaged in execution another process.

5)From ready state this process goes to running state and prints i value

6)Here process execution completes and it terminates

```
#include<stdio.h>
main()
{
 int i;
 printf("Hi...\n");
 scanf("%d",&i);
printf("i=%d\n",i);
}
```

States of Process (Contd..)


New
(1)
Admitted
Ready
(4)
I/O or Event Completion
Wait
(2) (5)
Scheduler
Dispatch
(3)
I/O Wait
Interrupted
Running
Terminated
Exit (6)