

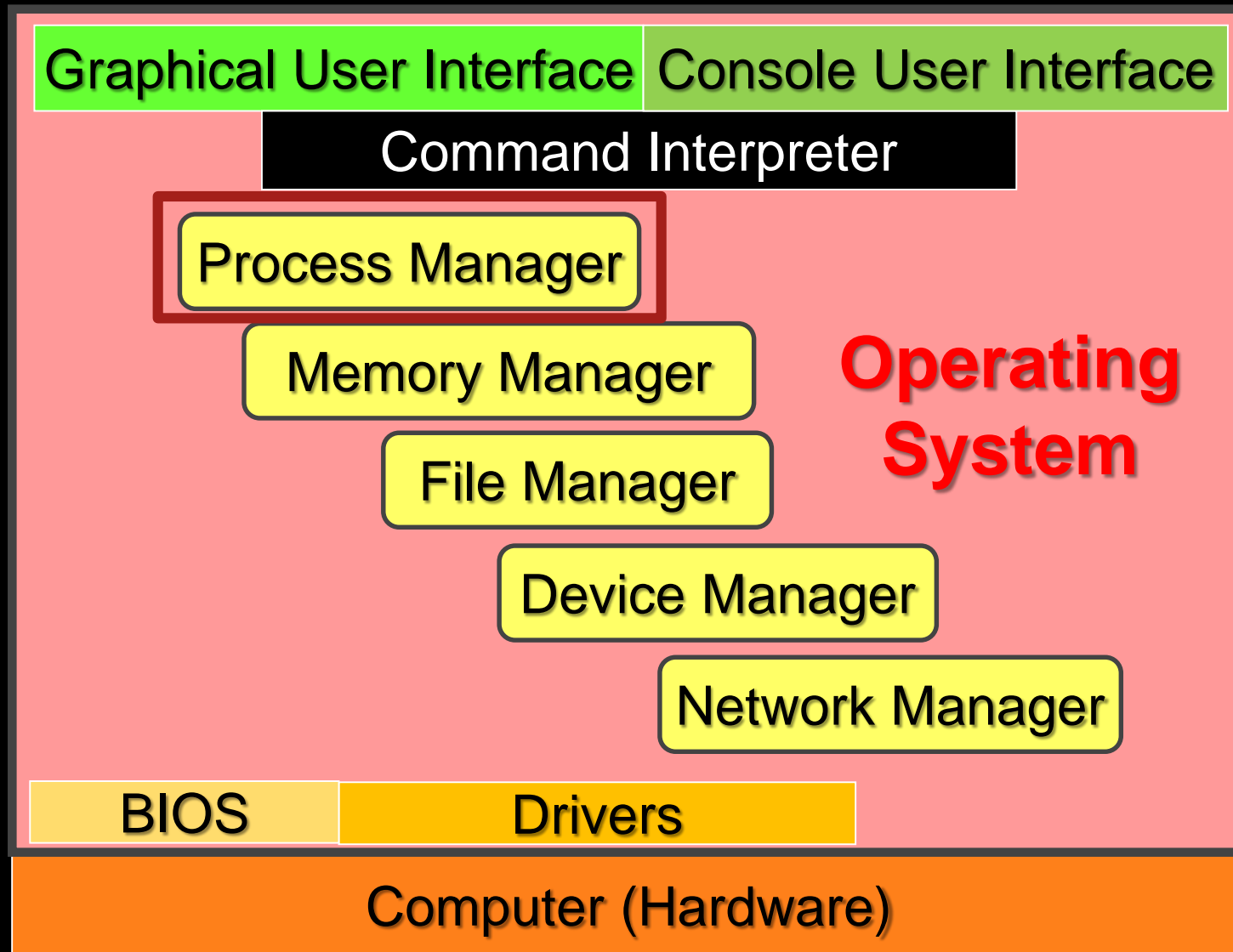
PROCESS MANAGEMENT

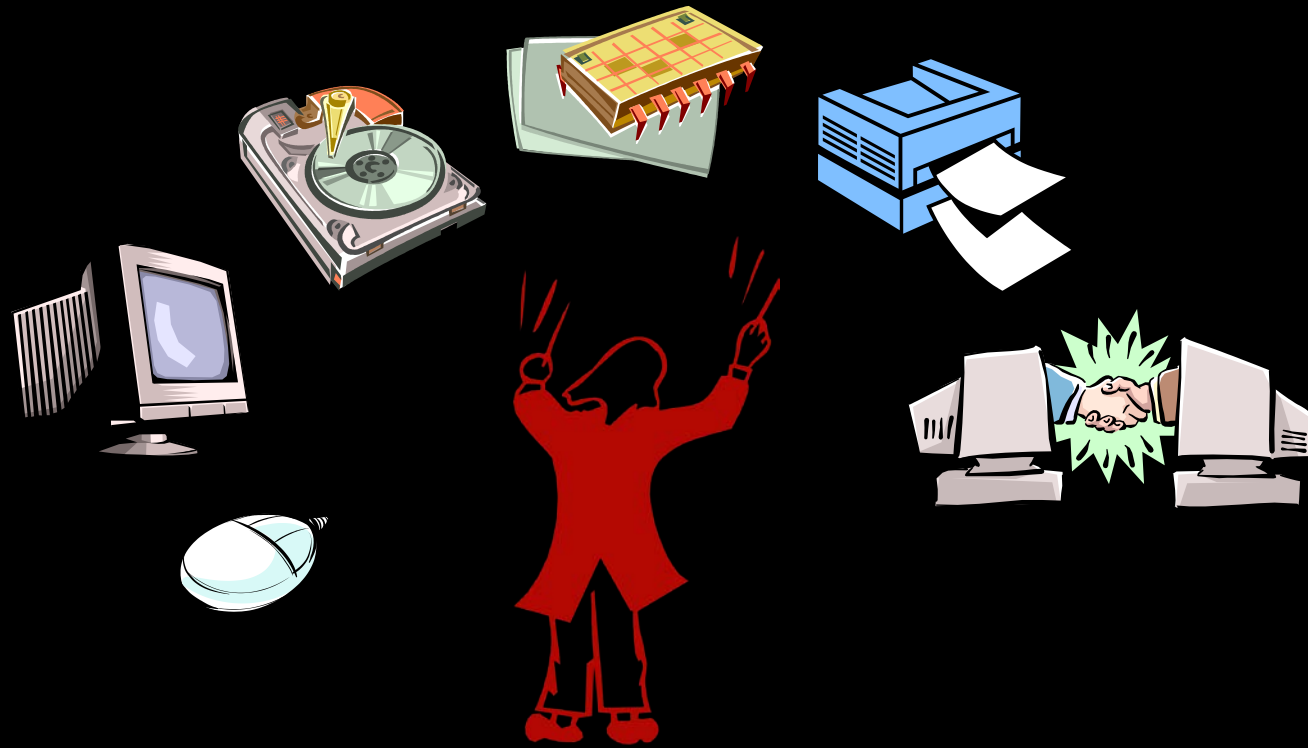
IT2758 Operating Systems

CONTENTS

- Programs/Processes/Threads
- Process States
- CPU Scheduling Schemes
- Single/Multicore Processing

OS ARCHITECTURE



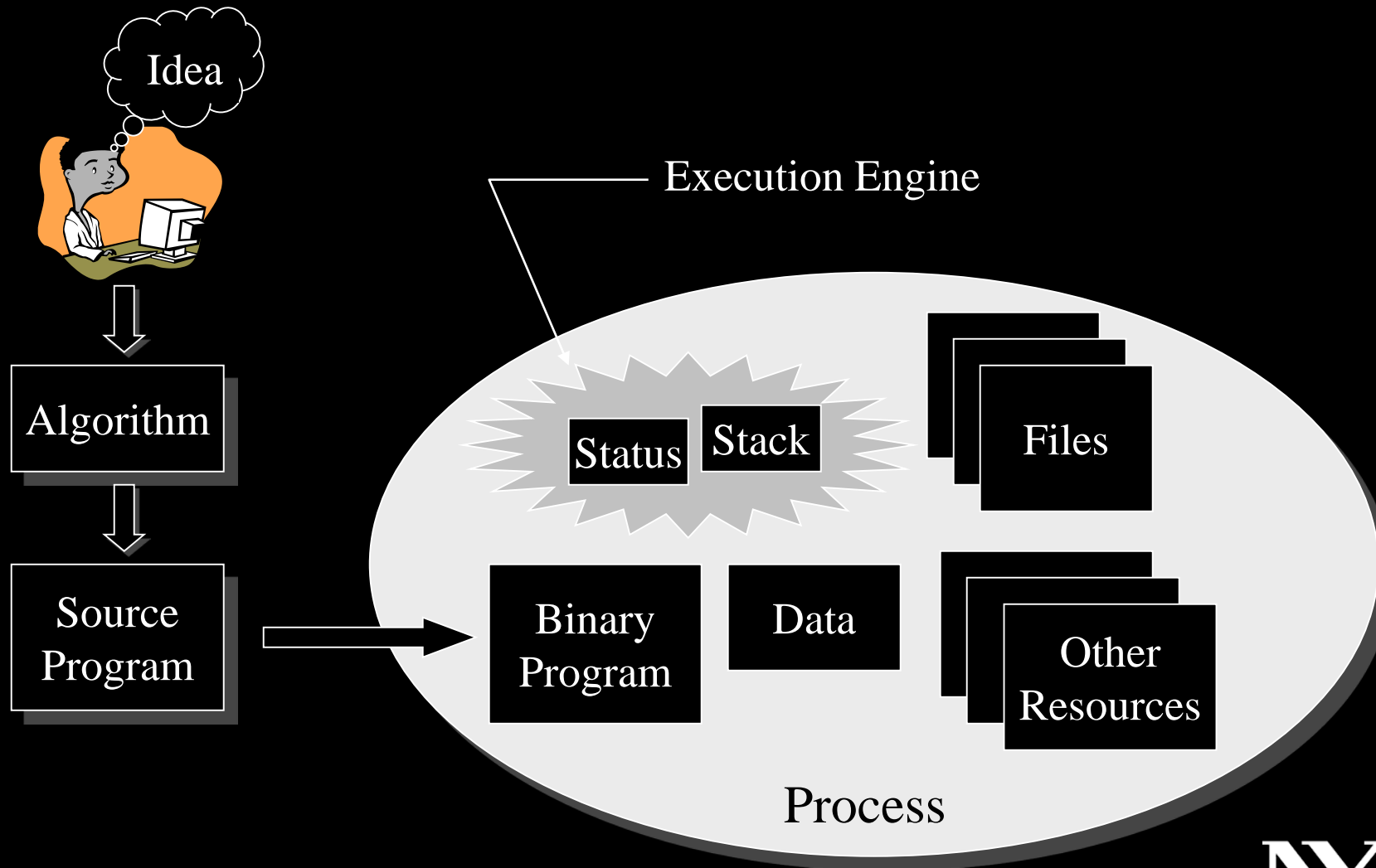


The OS coordinates the sharing and use of all the components (resources) in the computer

PROGRAMS/PROCESSES/THREADS

- Program is an image stored inside the disk
- Process is an instance of a program
 - A binary program in execution
 - May include data on which the program will execute
 - Resources required for execution, including files, devices which contains or provides the data required
 - Each process has a unique ID issued by Process Manager
- Resource: Anything that is needed for a process to run
 - Memory
 - Disk Space
 - CPU

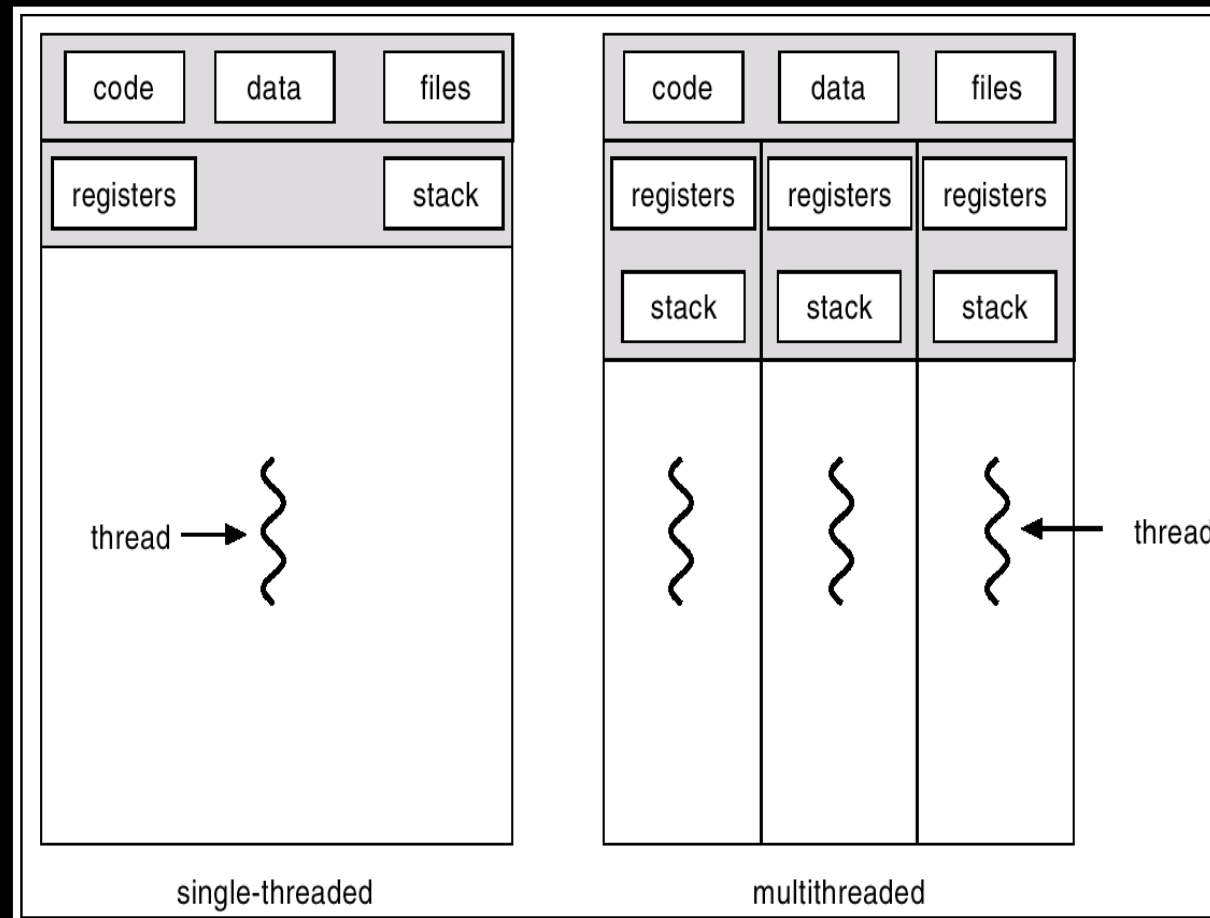
PROGRAMS/PROCESSES/THREADS



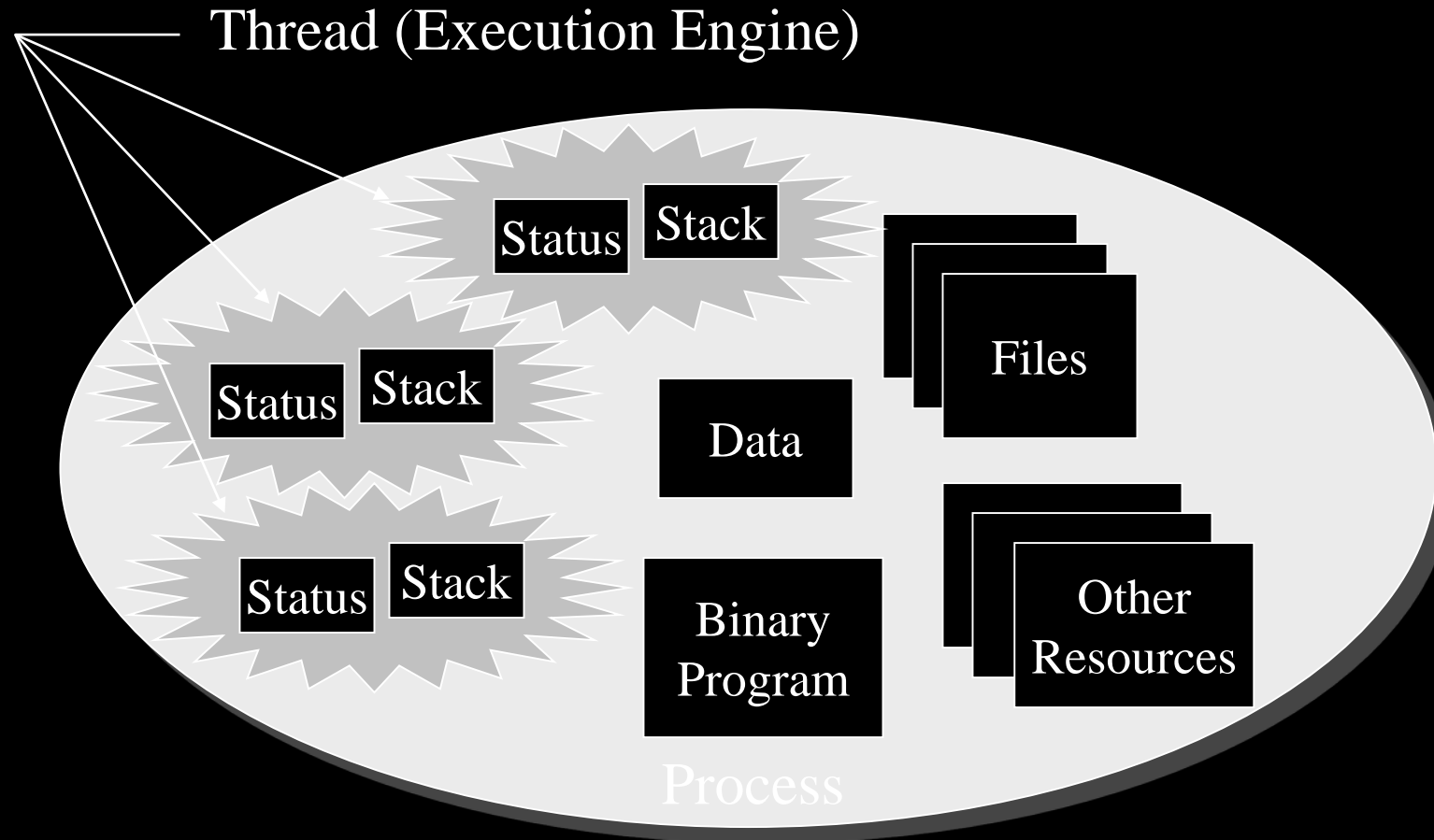
PROGRAMS/PROCESSES/THREADS

- Threads are lightweight sub-processes within a process
 - Program counter
 - Status of the thread
 - Processor registers
 - Stack space
- Threads within the same process shares the same :
 - Program code
 - Data
 - Resources

PROGRAMS/PROCESSES/THREADS

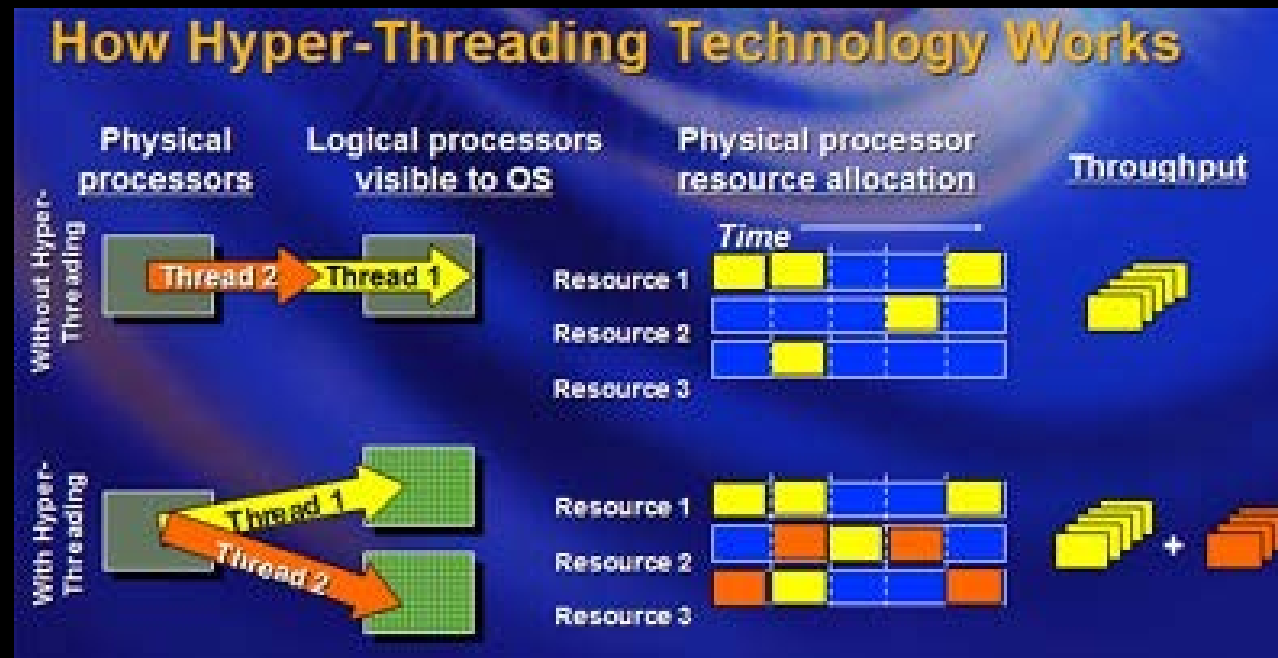


PROGRAMS/PROCESSES/THREADS



INTEL HYPER-THREADING

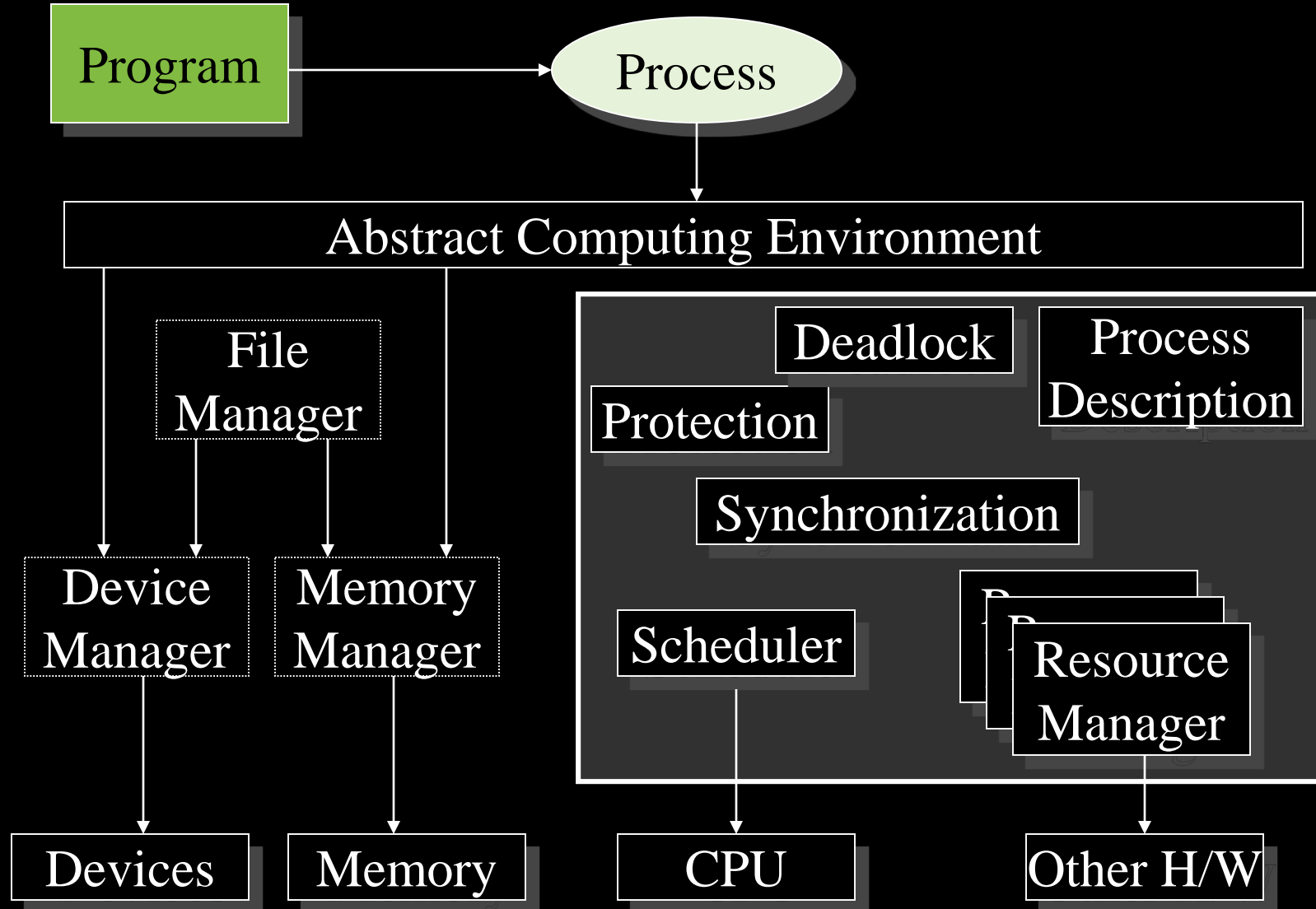
- A technology that presents a single core CPU as multiple logical cores (virtual CPUs).
- Not all operating systems support this feature.



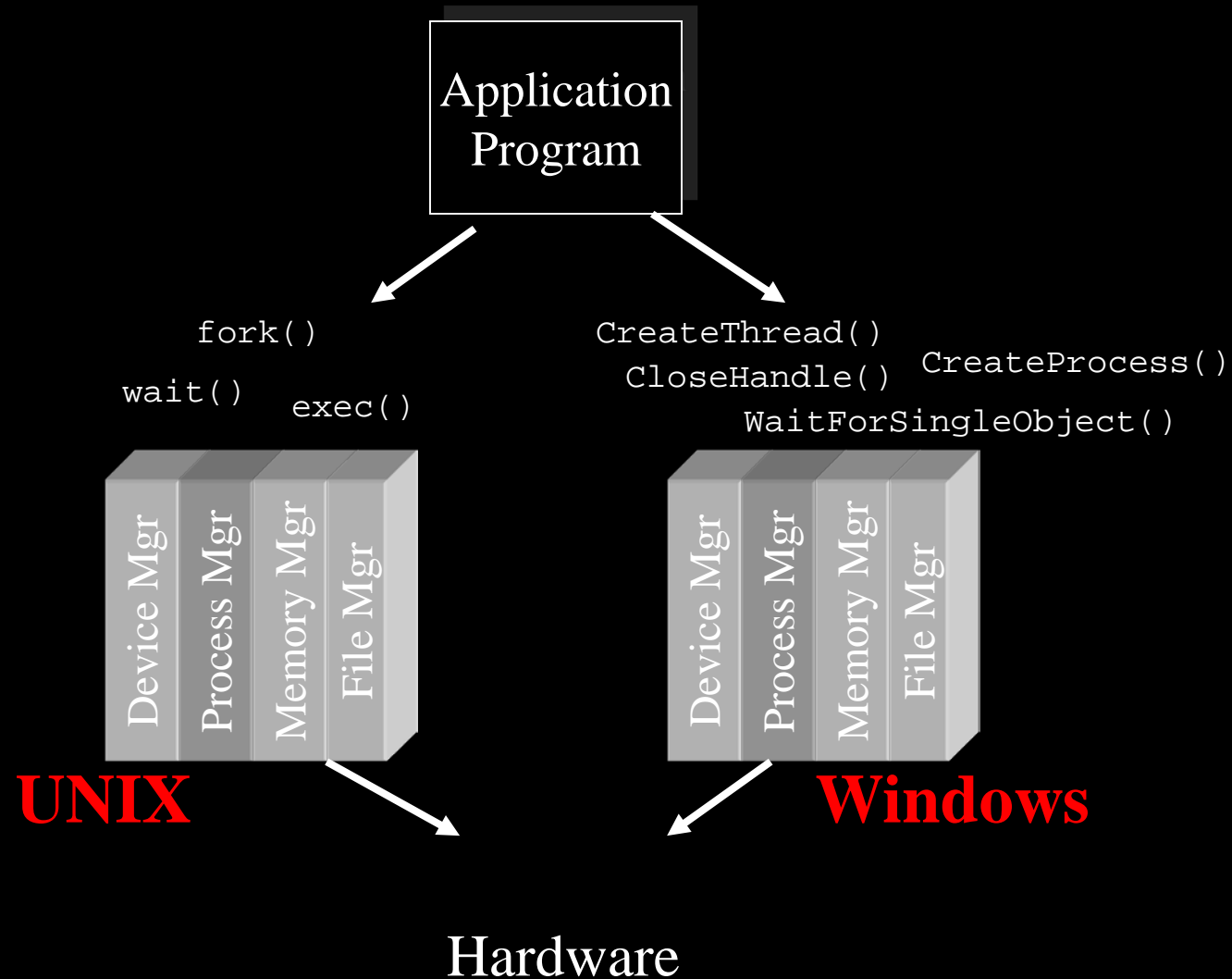
PROCESS MANAGER

- To manage multiple processes, modern OS implement the process manager to manage the processes.
- The process manager implements :
 - Process ID allocation and tracking
 - Calls like `fork()` in UNIX and `CreateProcess()` in windows to create processes.
 - Calls like `pthread_create()` in Linux and `CreateThread()` in Windows to support threading.
 - Calls like `close()` in Unix and `CloseHandle()` in Windows to close processes/threads to release resources.

PROCESS MANAGER OVERVIEW



EXTERNAL VIEW OF THE PROCESS MANAGER



UNIX PROCESSES

- OS kernel creates a process descriptor to manage process
- Process identifier (PID): User handle for the process (descriptor)
- Shell command to display processes:
 - `ps`
 - `ps -ef`
 - `ps aux`
 - `top`
- Classic Unix processes have not explicit notion of a thread.

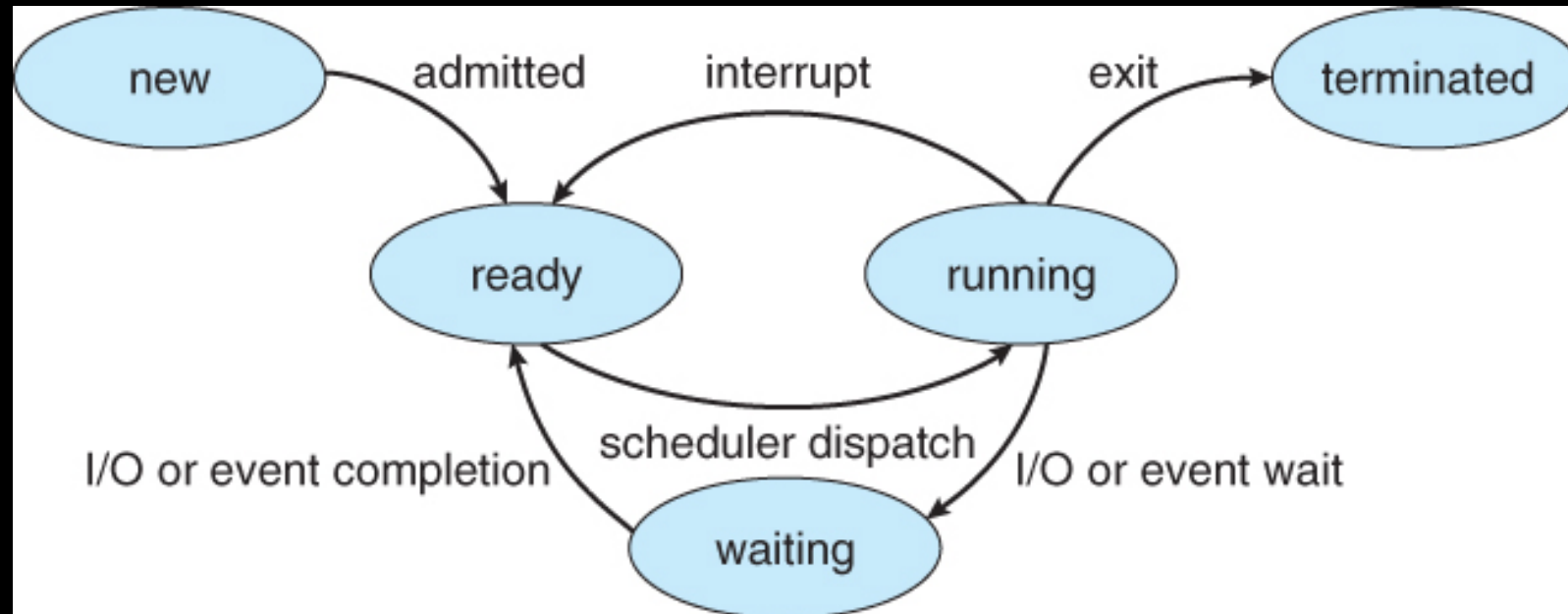
WINDOWS PROCESSES

- Windows Win32 API allows processes with multiple threads to be created through its CreateProcess() function.
- Options provided include :
 - Creating a new child process with a single thread.
 - Creating new additional threads in the current process.
- SysInternals
 - Add-ons to system applications
 - pslist
 - pskill

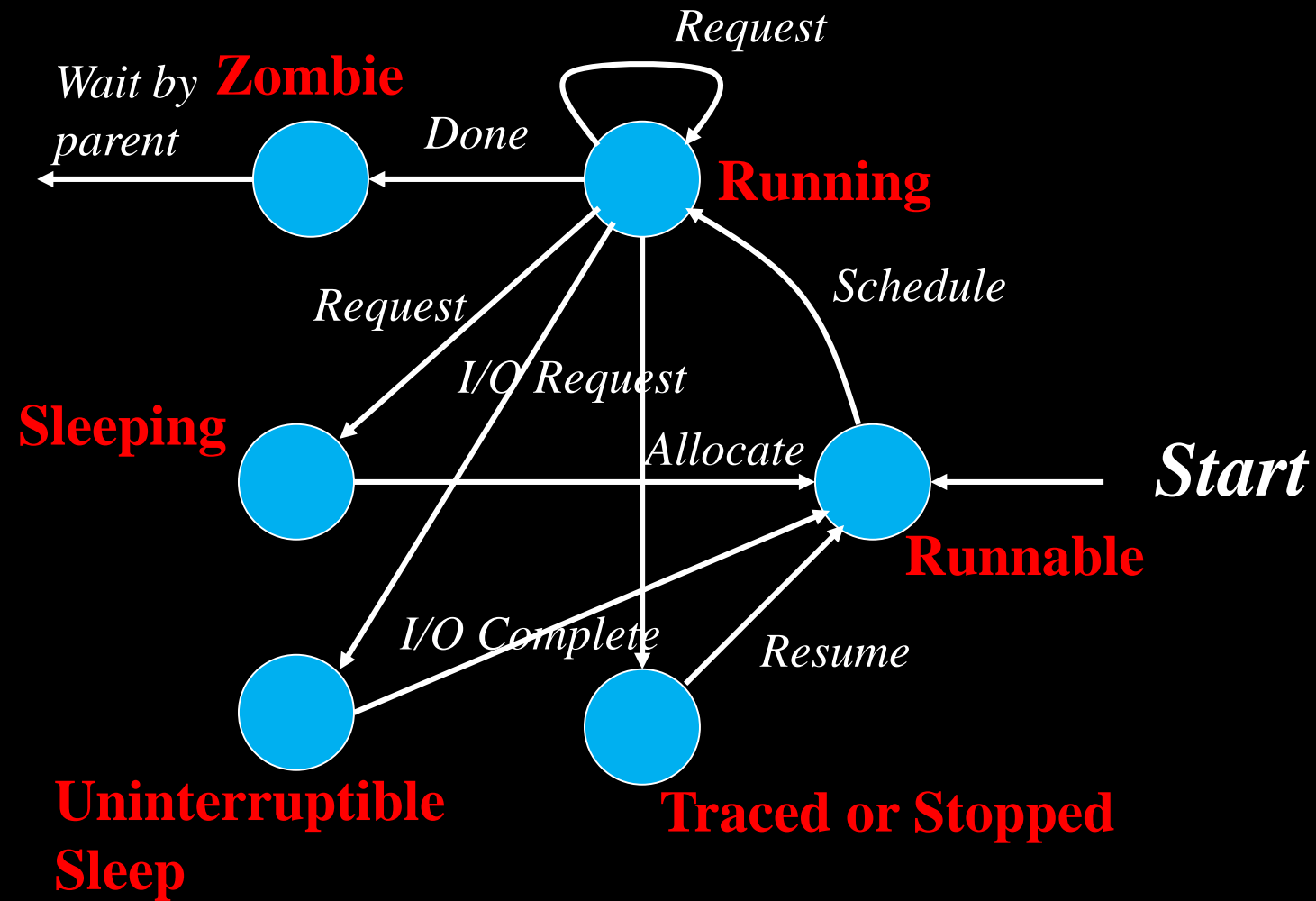
PROCESS STATES

- As a process executes, it changes state:
 - Running: Instructions are being executed
 - Waiting: The process is waiting for some event to occur (eg, I/O completion)
 - Ready: The process is waiting to be assigned to a processor.
 - Terminated: The process has finished execution
- Modern OS implement additional states as required to support more complex features.

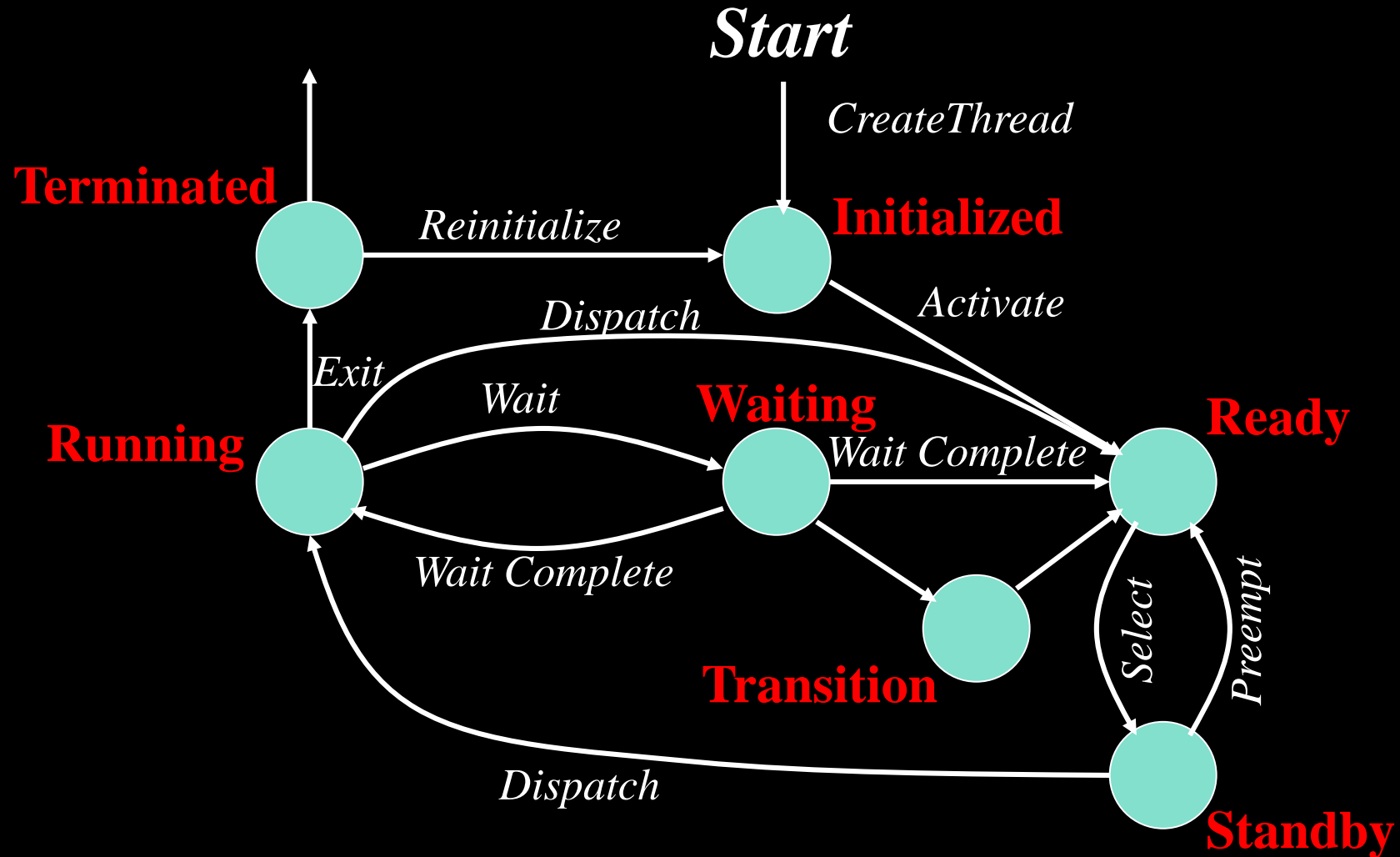
GENERIC PROCESS STATES



UNIX/LINUX PROCESS STATES



WINDOWS PROCESS STATES



SUPPORTING MULTIPLE PROCESSES

- Modern OS is required to support multiple processes (multi-tasking).
- In the case of a single physical CPU, there is a need to share the CPU among the many processes running at the same time.
 - Similar to how patients share a GP doctor in a clinic.
- There is a need to implement an abstraction mechanism to create an environment where multiple processes can execute.
- Each process executes as if it owns the CPU.

SUPPORTING MULTIPLE PROCESSES

- To ensure efficient use of CPU resources, CPU scheduling algorithms are needed
- Goals of scheduling - To achieve:
 - High processor utilization
 - High throughput
 - number of processes completed per unit time
 - Low response time
 - time elapsed from the submission of a request to the beginning of the first response

Non Pre-emptive

FIRST COME FIRST SERVE (FCFS)

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1, P_2, P_3
The Gantt Chart for the schedule is:



- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

Non Pre-emptive

SHORTEST JOB FIRST (SJF)

<u>Process</u>	<u>Burst Time</u>
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



- Average waiting time = $(3 + 16 + 9 + 0) / 4 = 7$

Non Pre-emptive

PRIORITY SCHEDULING

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart



- Average waiting time = 8.2 msec

Pre-emptive

ROUND ROBIN (RR)

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better *response*
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

MULTICORE PROCESSING

- Single core CPU
 - Execute one process at one time
- Single core + Hyper Threading CPU
 - Still single core physically
 - Fools the operating system that there are more cores logically
 - Execute multiple process at one time
- Multi cores CPU
 - Each core is a single CPU
 - Execute multiple processes at one time
- Nowadays, we have multicores + hyper-threading !!!