[03] PROCESSES

OUTLINE

- Process Concept
  - Relationship to a Program
  - What is a Process?
- Process Lifecycle
  - Creation
  - Termination
  - Blocking
- Process Management
  - Process Control Blocks
  - Context Switching
  - Threads

PROCESS CONCEPTS

- **Process Concept**
  - **Relationship to a Program**
  - **What is a Process?**
- Process Lifecycle
- Process Management

WHAT IS A PROCESS?

The computer is there to execute programs, not the operating system!

Process ≠ Program

- A program is **static**, on-disk
- A process is **dynamic**, a program *in execution*

On a batch system, might refer to *jobs* instead of processes

WHAT IS A PROCESS?

Unit of protection and resource allocation

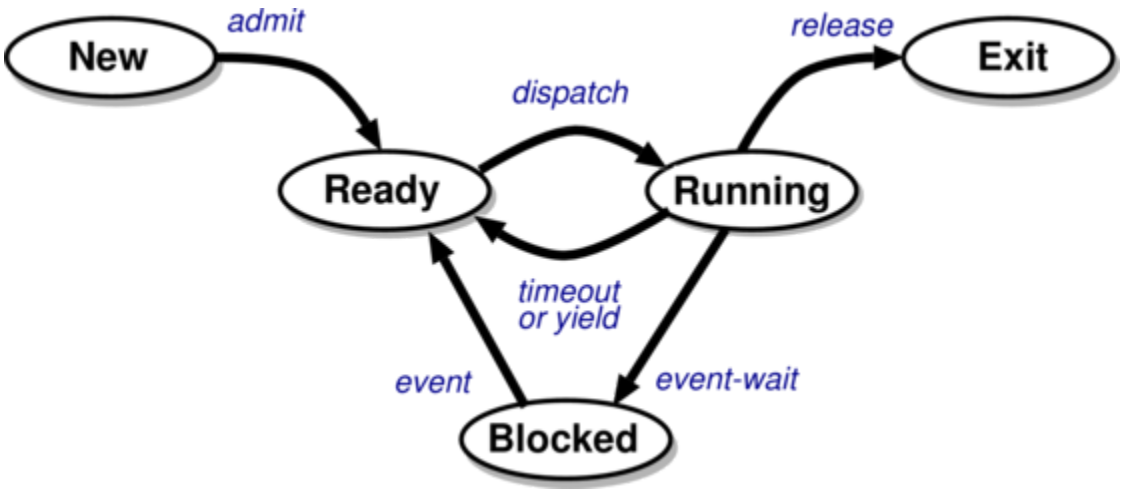
- So you may have multiple copies of a process running
- Each process executed on a *virtual processor*

Has a virtual address space (later)

Has one or more threads, each of which has

1. **Program Counter**: which instruction is executing
2. **Stack**: temporary variables, parameters, return addresses, etc.
3. **Data Section**: global variables shared among threads

PROCESS STATES



- **New**: being created
- **Running**: instructions are being executed
- **Ready**: waiting for the CPU, ready to run
- **Blocked**: stopped, waiting for an event to occur
- **Exit**: has finished execution

PROCESS LIFECYCLE

- Process Concept
- **Process Lifecycle**
  - **Creation**
  - **Termination**
  - **Blocking**
- Process Management

PROCESS CREATION

Nearly all systems are hierarchical:
parent processes create child processes

- Resource sharing:
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources

PROCESS CREATION

Nearly all systems are hierarchical:
parent processes create child processes

- Resource sharing
- Execution:
  - Parent and children execute concurrently
  - Parent waits until children terminate

PROCESS CREATION

Nearly all systems are hierarchical:
parent processes create child processes

- Resource sharing
- Execution
- Address space:
  - Child duplicate of parent
  - Child has a program loaded into it

EXAMPLES

Unix:

- `fork()` system call creates a child process, cloned from parent; then
- `execve()` system call used to replace the process' memory space with a new program

NT/2K/XP:

- `CreateProcess()` system call includes name of program to be executed

PROCESS TERMINATION

Occurs under three circumstances

1. Process executes last statement and asks the operating system to delete it (exit):
    - Output data from child to parent (wait)
    - Process' resources are deallocated by the OS

PROCESS TERMINATION

Occurs under three circumstances

1. Process executes last statement and asks the operating system to delete it
2. Process performs an illegal operation, e.g.,
    - Makes an attempt to access memory to which it is not authorised
    - Attempts to execute a privileged instruction

PROCESS TERMINATION

Occurs under three circumstances

1. Process executes last statement and asks the operating system to delete it
2. Process performs an illegal operation
3. Parent may terminate execution of child processes (abort, kill), e.g. because
    - Child has exceeded allocated resources
    - Task assigned to child is no longer required
    - Parent is exiting ("cascading termination")

EXAMPLES

Unix

- `wait()`, `exit()` and `kill()`

NT/2K/XP

- `ExitProcess()` for self
- `TerminateProcess()` for others.

BLOCKING

- In general a process blocks on an event, e.g.,
  - An IO device completes an operation
  - Another process sends a message
- Assume OS provides some kind of general-purpose blocking primitive, e.g., `await()`
- Need care handling concurrency issues, e.g.,

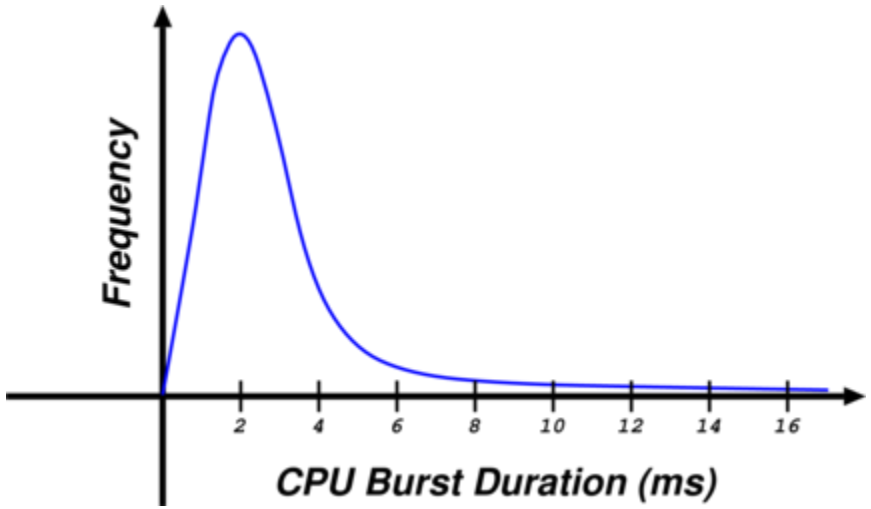
```
if(no key being pressed) {
  await(keypress);
  print("Key has been pressed!\n");
}
// handle keyboard input
```

  - What happens if a key is pressed at the first `{`?
  - Complicated! Next year... Ignore for now :)

CPU IO BURST CYCLE

- Process execution consists of a cycle of CPU execution and IO wait
- Processes can be described as either:
  1. **IO-bound**: spends more time doing IO that than computation; has many short CPU bursts
  2. **CPU-bound**: spends more time doing computations; has few very long CPU bursts

CPU IO BURST CYCLE



Observe that most processes execute for at most a few milliseconds before blocking

We need multiprogramming to obtain decent overall CPU utilisation

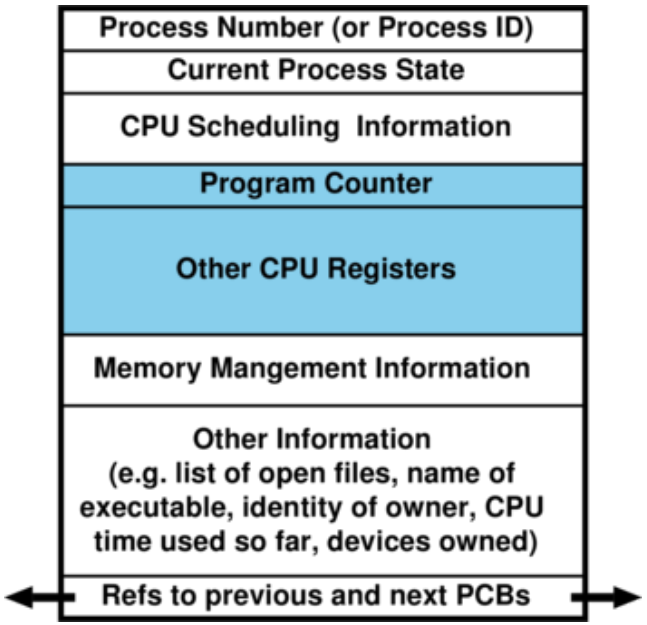
PROCESS MANAGEMENT

- Process Concept
- Process Lifecycle
- **Process Management**
  - **Process Control Blocks**
  - **Context Switching**
  - **Threads**

PROCESS CONTROL BLOCK

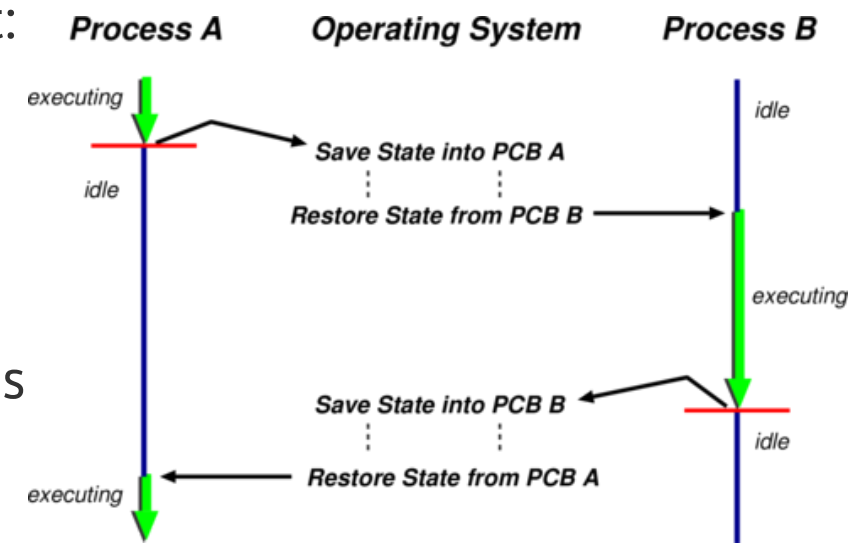
OS maintains information about every process in a data structure called a *process control block* (PCB). The *Process Context* (highlighted) is the machine environment during the time the process is actively using the CPU:

- Program counter
- General purpose registers
- Processor status register
- [Caches, TLBs, Page tables, ...]



CONTEXT SWITCHING

- To switch between processes, the OS must:
  - Save the context of the currently executing process (if any), and
  - Restore the context of that being resumed.
- Note this is *wasted time* — no useful work is carried out while switching
- Time taken depends on hardware support
  - From nothing, to
  - Save/load multiple registers to/from memory, to
  - Complete hardware "task switch"



THREADS

A **thread** represents an individual execution context

Threads are managed by a **scheduler** that determines which thread to run

Each thread has an associated **Thread Control Block** (TCB) with metadata about the thread: saved context (registers, including stack pointer), scheduler info, etc.

Context switches occur when the OS saves the state of one thread and restores the state of another. If between threads in different processes, process state also switches

Threads visible to the OS are **kernel threads** — may execute in kernel or address user space

SUMMARY

- Process Concept
  - Relationship to a program
  - What is a process?
- Process Lifecycle
  - Creation
  - Termination
  - Blocking
- Process Management
  - Process Control Blocks
  - Context Switching
  - Threads