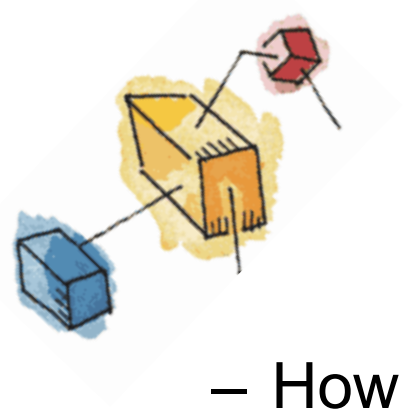
Operating Systems:
Internals and Design Principles
William Stallings

Process d Control



Objectives

- How are processes represented and controlled by the OS.
- ***Process states*** which characterize the behaviour of processes.
- ***Data struct*** processes.
- Ways in which the OS use a structures to control process execution.

Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users he system through terminals

Process

- Fundamental to the structure of operating systems

A *p* as:

a program in e

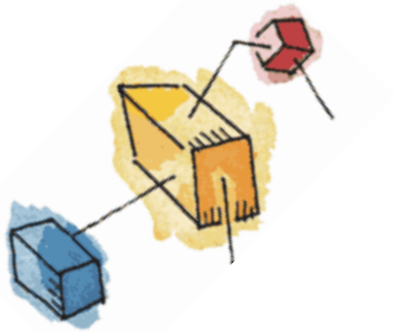
an instance of a running program

the entity that can be assigned to, and executed on, a processor

a unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

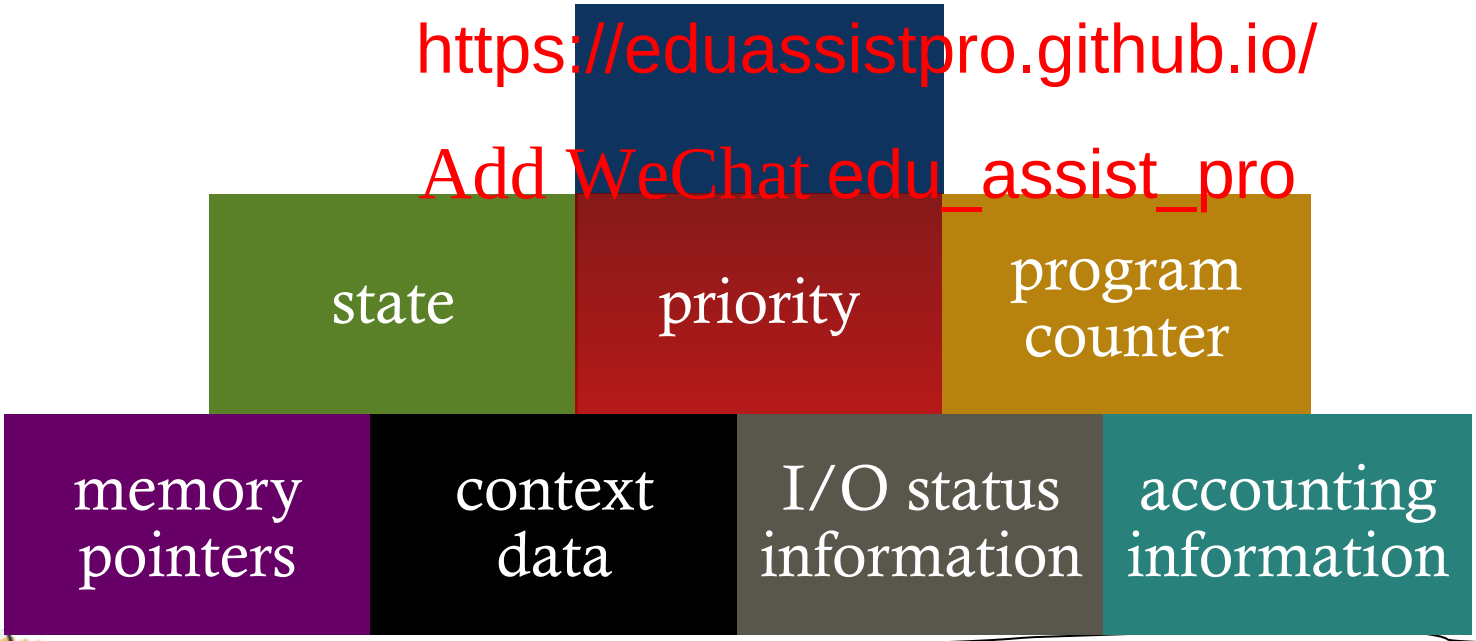
Process Management

- Is the *Fundamental Task*
- The Operating System must
  - Allocate resources to processes, and protect the resources o er processes,
  - Interleave t processes
  - Enable processes to shar ge information,
  - Enable synchronization among processes.



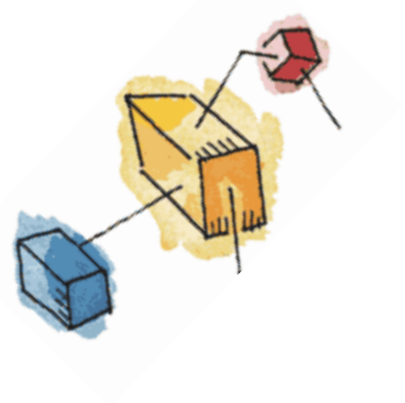
Process Elements

- While the program is executing, this process can be uniquely characterized by a number of *attributes*, including:

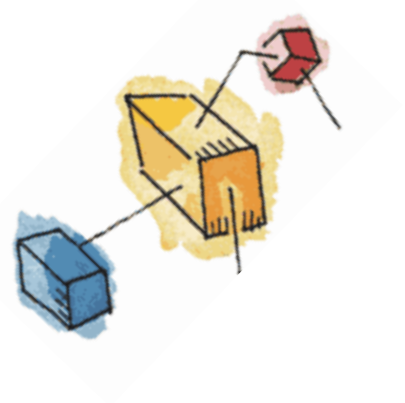


Process Control Block

- The most important data structure in an OS
- Contains the process attributes
- Created and m g system
- Key tool that allows support f processes
- Attributes in general categories:
  - Process identification
  - Processor state information
  - Process control information



Process Attributes

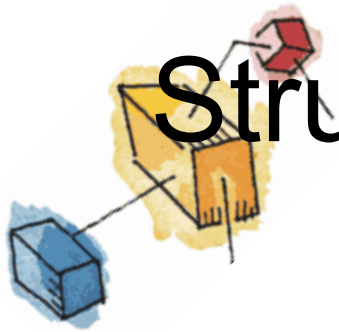


Process Attributes

Process Image

- Typical elements:

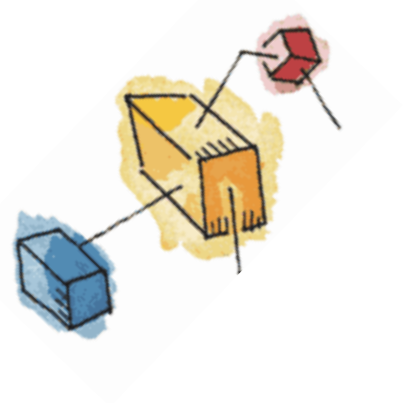
- Process image location will depend on the memory management scheme being used



Structure of Process Images in Virtual Memory

Operating System Control Structures

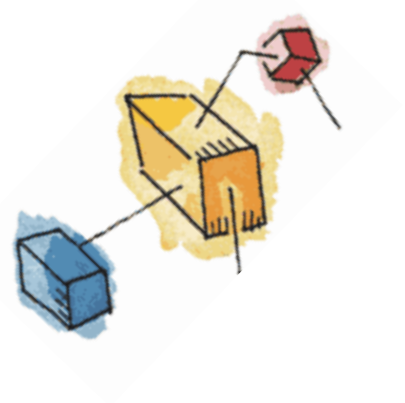
- For the OS to manage processes and resources, it must have information about the current status of each process and resource
- Tables are con (memory, I/O and files) the opera



OS Control Tables

Process Tables

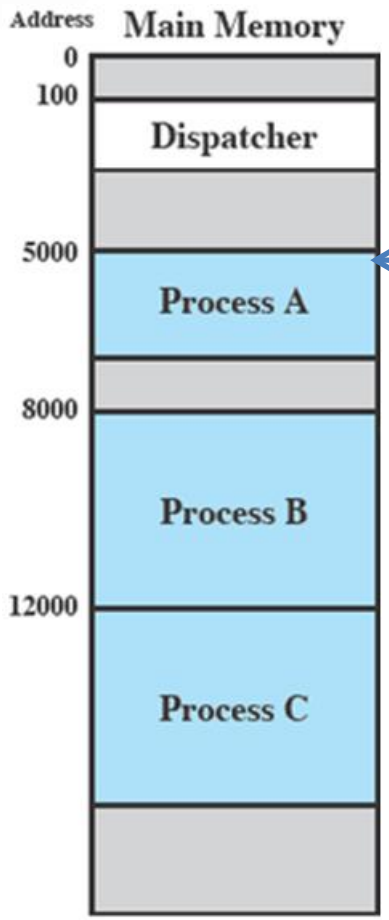
- Must be maintained to manage processes
- The OS tables must be linked or cross-referenced
  - Memory, I/O and files are managed on behalf of processes, es to these resources, he process tables.

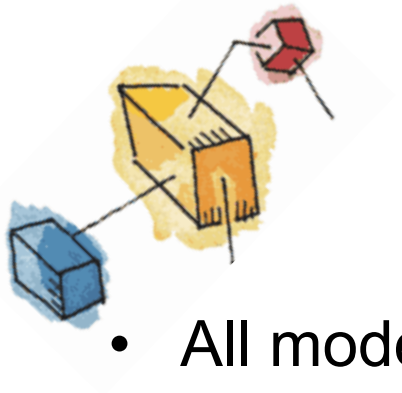


Process Execution

- ***Dispatcher*** is a small program which switches the processor from one process to another
- Several processes are started
- All are in memory (plus the dispatcher)

Trace from Processors point of view



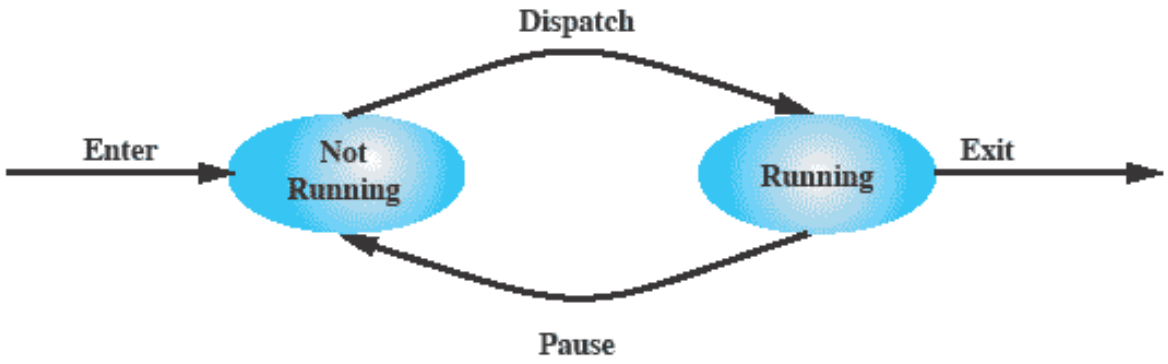


Modes of Execution

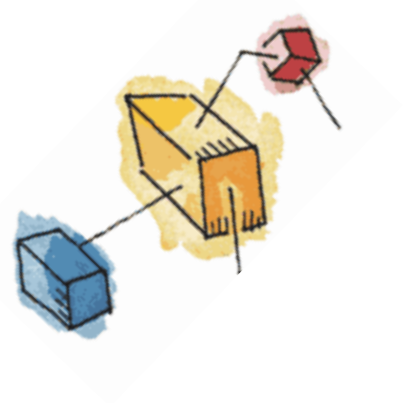
- All modern processors support at least two modes of execution
- User mode
  - Less-privile
  - User progra this mode
- System (or kernel) mode
  - More-privileged mode
  - Kernel of the operating system

Two-State Process Model

- The state of a process may be defined by the current activity of that process
  - Used to describe the behaviour that we would like each process to exhibit
- Process may b
  - Running
  - Not-running

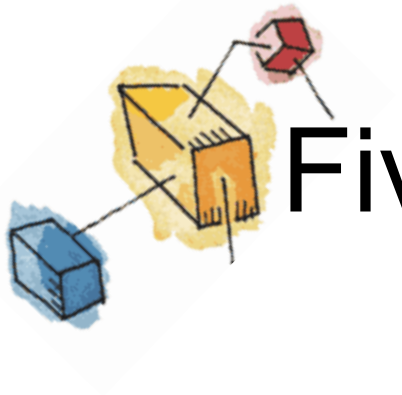


(a) State transition diagram



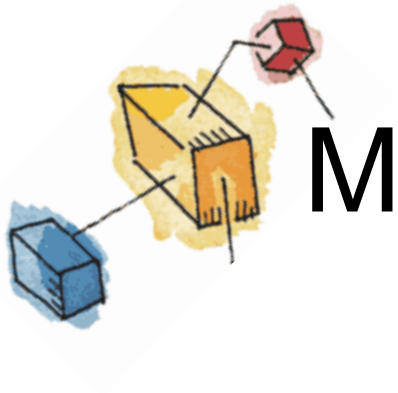
Queuing Diagram

Etc … processes moved by the dispatcher of the OS to the CPU then back to the queue until the task is completed



Five-State Process Model

Using Two Queues



Multiple Blocked Queues

Adding Suspend States

Process List Structures

Switching Processes

- Several design issues are raised regarding process switching
  - What events trigger a process switch?
  - What must t data structures under its co ss switch?

When to switch processes

A process switch may occur any time that the OS has gained control from the currently running process. Possible events giving OS control are:

Mechanism	Cause	Use
Interrupt	Exter the c	tion to an asynchronous rnal event
Trap	Associated with the exec of the current instruction	g of an error or an exception condition
Supervisor call	Explicit request	Call to an operating system function

Table 3.8 Mechanisms for Interrupting the Execution of a Process

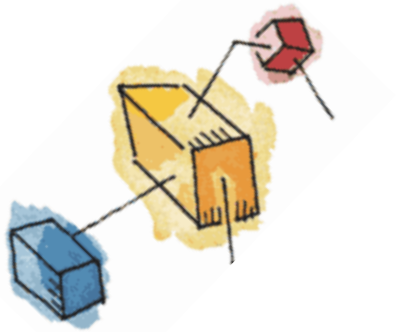
System Interrupts

- Interrupt
  - Due to some sort of event that is external to and independent of the currently running process
    - clock interrupt
    - I/O interrupt
  - Time slice
    - the maximum execute before being interrupted
- Trap
  - An error or exception condition generated within the currently running process
  - OS determines if the condition is fatal
    - moved to the Exit state and a process switch occurs
    - action will depend on the nature of the error

Change of Process State

- The steps in a process switch are:

If the currently runni
another state (Ready, Blocked, etc.),
must make substantial changes in its



Process Creation

- Once the OS decides to create a new process it:

Process Creation (cont.)

- Traditionally, the OS created all processes
  - But it can be useful to let a running process create another
- This action is c ***g*** ting, process
  - ***Parent Pro***
  - ***Child Process*** is the new
- Parent process create children processes, which, in turn create other processes, forming a tree of processes

Process Creation (cont.)

- A Tree of Processes in UNIX/Linux:

Process Creation (cont.)

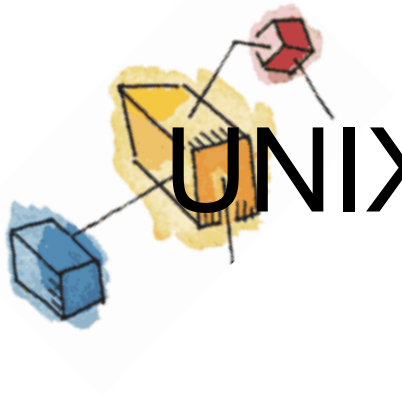
- Generally, process identified and managed via **a process identifier** (**pid**)
- Resource sharing
  - Parent and rces
  - Children sh sources
  - Parent and child share no
- Execution
  - Parent and children execute concurrently
  - Parent waits until children terminate

UNIX Process Creation

- Process creation is by means of the kernel system call, fork( ).
- This causes the OS, in Kernel Mode, to:
  1. Allocate a sl new process.
  2. Assign a uni rocess.
  3. Copy of process image of the he exception of any shared memory.
  4. Increment the counters for any files owned by the parent, to reflect that an additional process now also owns those files.
  5. Assign the child process to the Ready state.
  6. Returns the ID number of the child to the parent process, and a 0 value to the child process.

UNIX Process Creation (cont.)

- After creating the process the Kernel can do one of the following, as part of the dispatcher routine:
  - Stay in the parent process.
  - Transfer co
  - Transfer co



UNIX Process Creation (cont.)

Process Creation (cont.)

```
int main()
{
pid_t  pid;
    /* fork another process */
    pid = fork();
    if (pid < 0) { /* error occ
        fprintf(stderr, "Fo
        exit(-1);
    }
    else if (pid == 0) { /* child process */
        execlp("/bin/ls", "ls", NULL);
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait (NULL);
        printf ("Child Complete");
        exit(0);
    }
}
```


Process Termination

- There must be some way that a process can indicate completion.
- This indication may be:
  - A HALT instruction terrupt alert to the OS.
  - A user action (e.g. log off, application)
  - A fault or error
  - Parent process terminating

Security Issues

- An OS associates a set of privileges with each process.
  - Highest level being administrator, supervisor, or root, access.
- A key security ny OS is to prevent anything (user g unauthorized privileges on the system.
  - Especially - from gaining r

Summary

- The principal function of the OS is to create, manage, and terminate processes
- The most fundamental concept in a modern OS is the process
- Process control block contains all of the information that is required for the OS to man s current state, resources allocat levant data
- The most important states are Rea and Blocked
  - The running process is the one that is c ecuted by the processor
  - A blocked process is waiting for the completion of some event
  - A running process is interrupted either by an interrupt or by executing a supervisor call to the OS