

Process Concept

Processes 1

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

The diagram shows a vertical stack of memory sections. At the top is the 'stack' section, which is shaded grey and has a downward-pointing arrow indicating it grows towards lower memory addresses. Below the stack is the 'heap' section, also shaded grey, with an upward-pointing arrow indicating it grows towards higher memory addresses. Below the heap are the 'data' and 'text' sections, both shaded grey. The top of the stack is labeled 'max' and the bottom of the text section is labeled '0'.

Computer Science Dept Va Tech August 2007
Operating Systems
©2003-07 McQuain

Process State

Processes 2

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 process
- **terminated**: The process has finished execution

The diagram shows five states in ovals: 'new', 'ready', 'running', 'waiting', and 'terminated'. Transitions are as follows: 'new' to 'ready' (labeled 'admitted'); 'ready' to 'running' (labeled 'scheduler dispatch'); 'running' to 'ready' (labeled 'interrupt'); 'running' to 'waiting' (labeled 'I/O or event wait'); 'waiting' to 'ready' (labeled 'I/O or event completion'); 'running' to 'terminated' (labeled 'exit').

Computer Science Dept Va Tech August 2007
Operating Systems
©2003-07 McQuain

Process Control Block (PCB)

Processes 3

Information associated with each process

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

process state
process number
program counter
registers
memory limits
list of open files
• • •

aka: *process descriptor*

Computer Science Dept Va Tech August 2007
Operating Systems
©2003-07 McQuain

CPU Switch From Process to Process

Processes 4

process P_0	operating system	process P_1
<div style="text-align: center;">executing</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 100px; margin: 5px auto;"></div> <div style="text-align: center;">idle</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 100px; margin: 5px auto;"></div> <div style="text-align: center;">executing</div>	<div style="margin-bottom: 10px;">interrupt or system call</div> <div style="background-color: #cccccc; padding: 5px; border: 1px solid black;">save state into PCB_0</div> <div style="text-align: center;">⋮</div> <div style="background-color: #cccccc; padding: 5px; border: 1px solid black;">reload state from PCB_1</div> <div style="margin-bottom: 10px;">interrupt or system call</div> <div style="background-color: #cccccc; padding: 5px; border: 1px solid black;">save state into PCB_1</div> <div style="text-align: center;">⋮</div> <div style="background-color: #cccccc; padding: 5px; border: 1px solid black;">reload state from PCB_0</div>	<div style="border-left: 1px solid black; border-right: 1px solid black; height: 100px; margin: 5px auto;"></div> <div style="text-align: center;">idle</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 100px; margin: 5px auto;"></div> <div style="text-align: center;">executing</div> <div style="border-left: 1px solid black; border-right: 1px solid black; height: 100px; margin: 5px auto;"></div> <div style="text-align: center;">idle</div>

Computer Science Dept Va Tech August 2007
Operating Systems
©2003-07 McQuain

Process Scheduling Queues

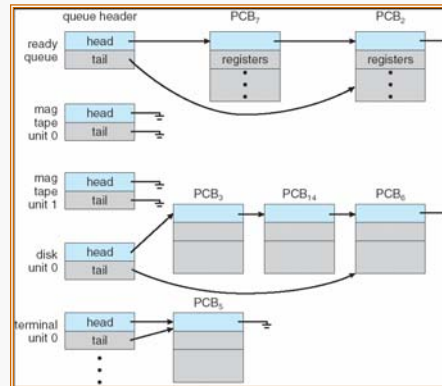
Processes 5

Job queue – set of all processes in the system

Ready queue – set of all processes residing in main memory, ready and waiting to execute

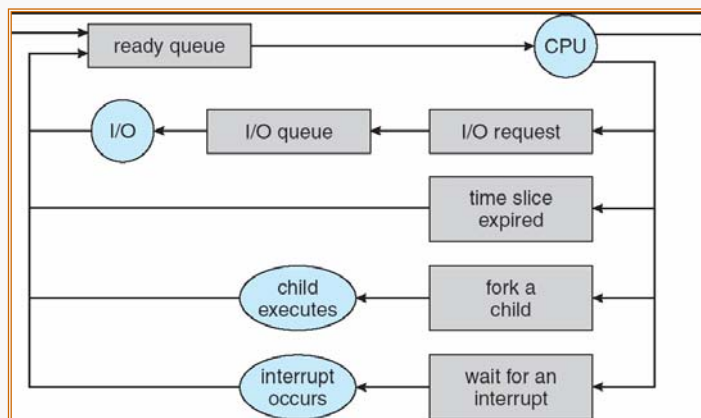
Device queues – set of processes waiting for an I/O device

Processes migrate among the various queues



Representation of Process Scheduling

Processes 6



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

Short-term scheduler is invoked very frequently (milliseconds) \Rightarrow (must be fast)

Long-term scheduler is invoked very infrequently (seconds, minutes) \Rightarrow (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

When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process

Context-switch time is overhead; the system does no useful work while switching

Time dependent on hardware support

Process Creation Processes 9

Parent process create children processes, which, in turn create other processes, forming a tree of processes

Resource sharing

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

Execution

- Parent and children execute concurrently
- Parent waits until children terminate

Address space

- Child duplicate of parent
- Child has a program loaded into it

UNIX examples

- **fork** system call creates new process
- **exec** system call used after a **fork** to replace the process' memory space with a new program

Computer Science Dept Va Tech August 2007 Operating Systems ©2003-07 McQuain

Process Creation Processes 10

```
graph LR; fork(fork()) -- parent --> wait(wait); fork(fork()) -- child --> exec(exec()); exec(exec()) --> exit(exit()); exit(exit()) --> wait(wait); wait(wait) -- resumes --> resume[ ];
```

Computer Science Dept Va Tech August 2007 Operating Systems ©2003-07 McQuain

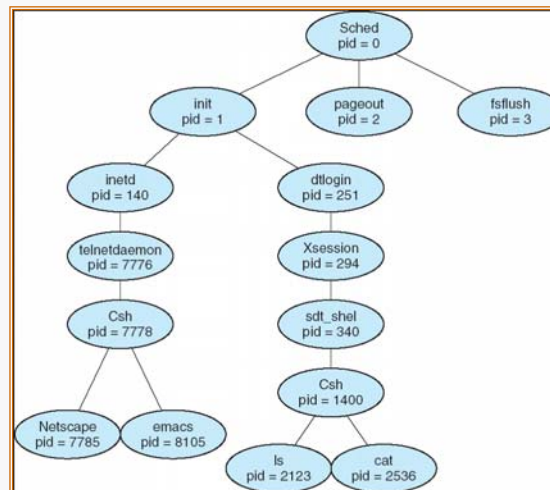
C Program Forking Separate Process

Processes 11

```
int main()
{
    pid_t pid;
    /* fork another process */
    pid = fork();
    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        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);
    }
}
```

A tree of processes on a typical Solaris

Processes 12



Process executes last statement and asks the operating system to delete it (**exit**)

- Output data from child to parent (via **wait**)
- Process' resources are deallocated by operating system

Parent may terminate execution of children processes (**abort**)

- Child has exceeded allocated resources
- Task assigned to child is no longer required
- If parent is exiting
 - Some operating system do not allow child to continue if its parent terminates
 - All children terminated - *cascading termination*