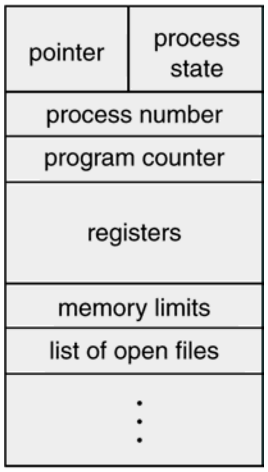
Processes

- Reading: OS Concepts pp.95-107
- A process is a program in execution and is represented by
  - ________________________
  - ________________________
  - ________________________
  - ________________________

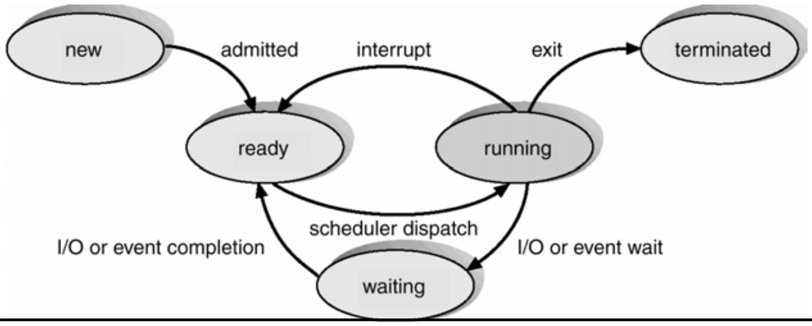
PCB

- Process Control Block contains
  - ________________
  - ________________
  - ________________
  - ________________
  - ________________
  - ________________

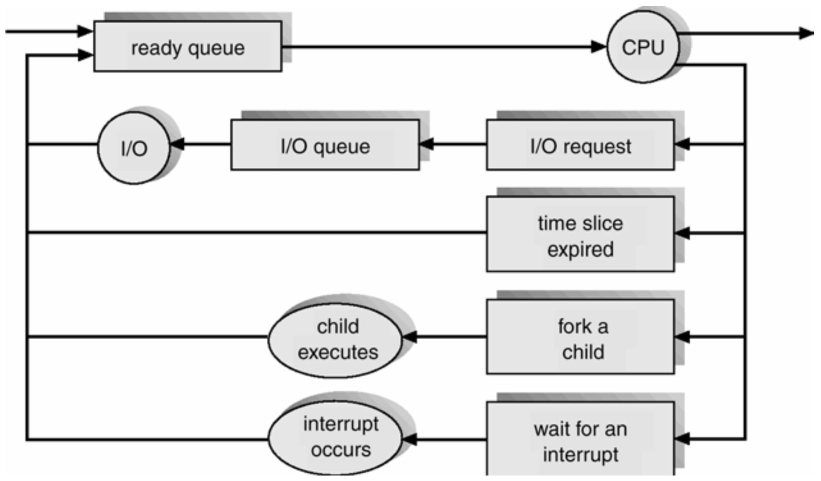


Process State

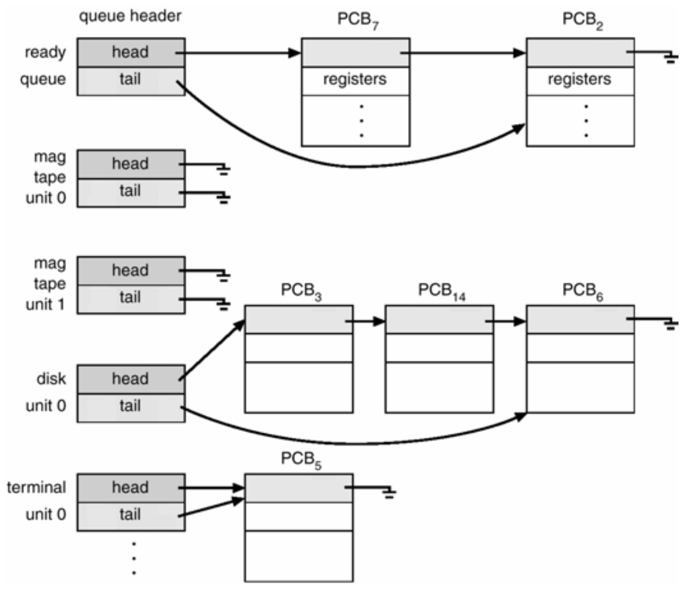
- New: ________________________________
- Ready: ________________________________
- Running: ______________________________
- Waiting: _______________________________
- Terminated: ____________________________



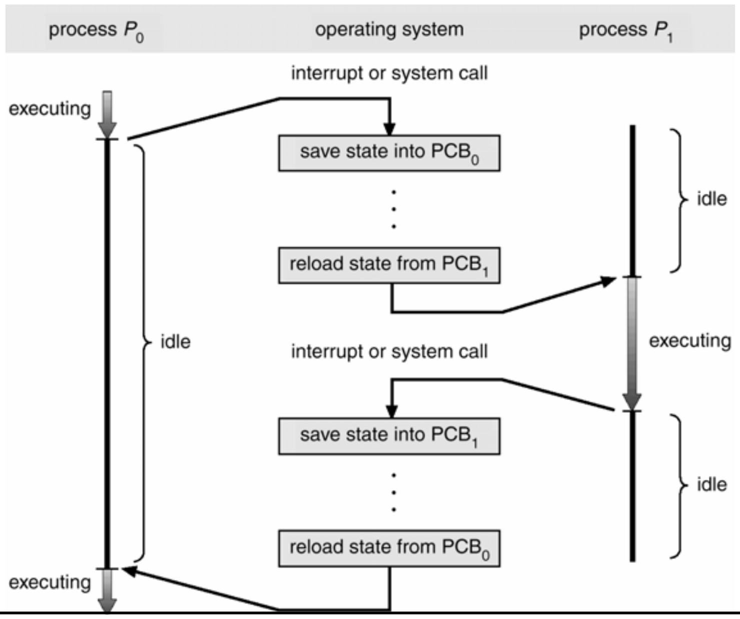
Process Scheduling



Scheduling Queues



Context Switch Illustrated



Process Management, Topic 1, Processes

Processes
Scheduling
Operations

Context Switch

- When the CPU switches between processes
  - ______________________________________
  - ______________________________________
  - ____ __________________________________
  - ______________________________________ ______________________________________
- Types of processes
  - I/O bound: ______________________________
  - CPU bound: ______________________________
  - ____

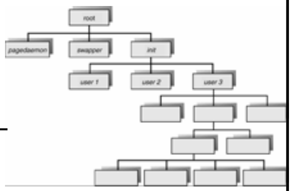
7

Process Management, Topic 1, Processes

Processes
Scheduling
Operations

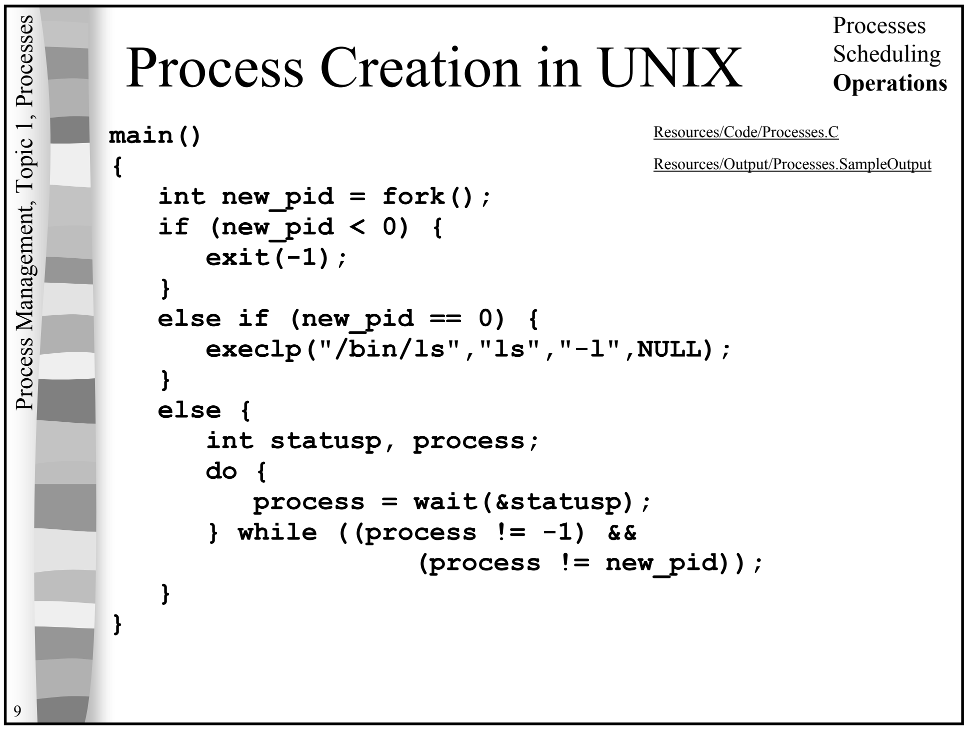
Operations – Creation

- Generally any process may create another process
- _____________________________
- _____________________________
- ______________________________ __________________________
- After creation the parent can either
  - ______________________________
  - or ____________________________
- And the child can either
  - __________________________________
  - or ____________________________



Typical UNIX process tree

8

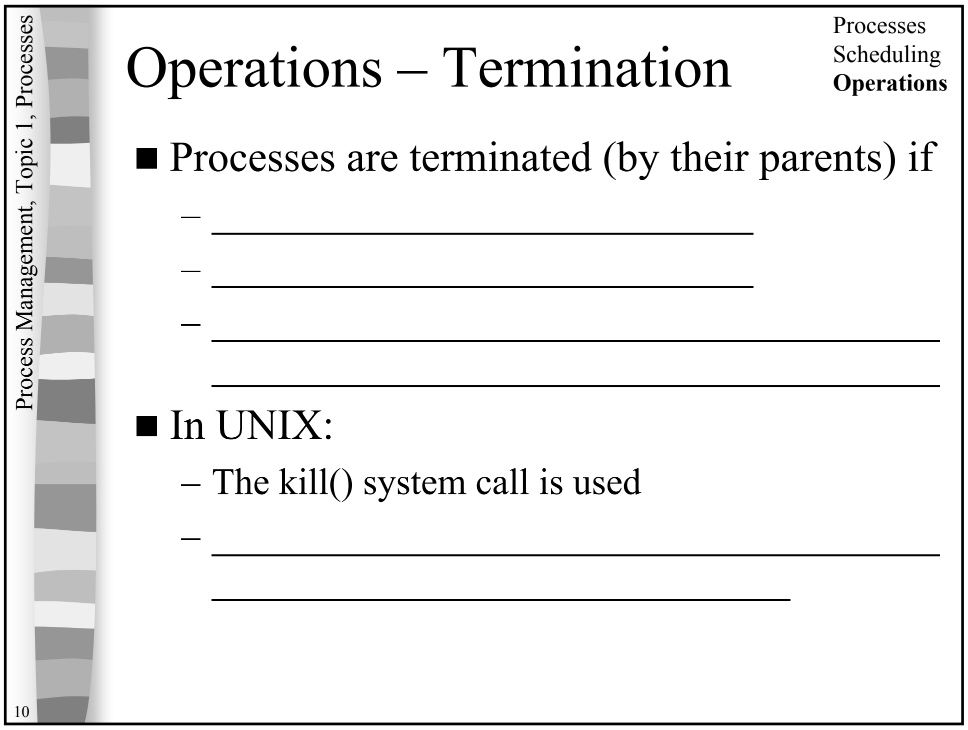


Process Creation in UNIX

Resources/Code/Processes.C

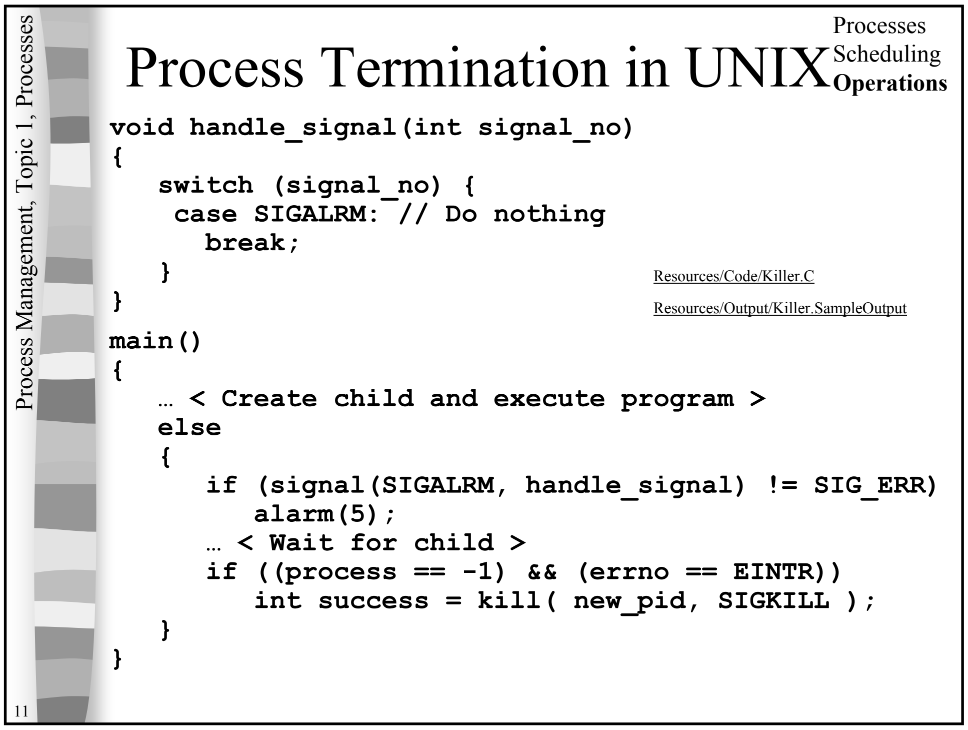
Resources/Output/Processes.SampleOutput

```
main()
{
   int new_pid = fork();
   if (new_pid < 0) {
      exit(-1);
   }
   else if (new_pid == 0) {
      execlp("/bin/ls","ls","-l",NULL);
   }
   else {
      int statusp, process;
      do {
         process = wait(&statusp);
      } while ((process != -1) &&
                      (process != new_pid));
   }
}
```



Operations – Termination

- Processes are terminated (by their parents) if
  - ______________________
  - ______________________
  - ______________________________
    ____
- In UNIX:
  - The kill() system call is used
  - ______________________________
    ____



Process Termination in UNIX

```
void handle_signal(int signal_no)
{
   switch (signal_no) {
    case SIGALRM: // Do nothing
      break;
   }
}
main()
{
   … < Create child and execute program >
   else
   {
      if (signal(SIGALRM, handle_signal) != SIG_ERR)
         alarm(5);
      … < Wait for child >
      if ((process == -1) && (errno == EINTR))
         int success = kill( new_pid, SIGKILL );
   }
}
```

Resources/Code/Killer.C

Resources/Output/Killer.SampleOutput