Processes

Chapter 2 of Tanenbaum

pseudoparallelism—rapid switching back and forth of the CPU between processes. Gives the illusion of parallelism. Make this process invisible to the user.

Figure 2.1, each process has its own program counter.

Operating System gives the illusion of executing multiple processes concurrently. At the heart of this process switching is

- *context switching* — stopping one process and starting another

Built on top of context context switching is

- *scheduling* — choosing a new process from those eligible for execution.

IMPORTANT: the operating system views processes as a set of data structures that can be manipulated.

Process States

```
cat chapter1 chapter2 chapter3 | grep tree
```

Will create two processes. Is possible for the `grep` process to be blocked waiting for input.

Look at Figure 2.2 in detail. Important.

Look at Figure 2.3.

Process Details

Process Table

The *process table* holds information about every process in the system. Silberschatz calls it the process control block (PCB).

Some of the entries are for saving the state of a blocked process. The information in each entry is called a *context block* and contains:

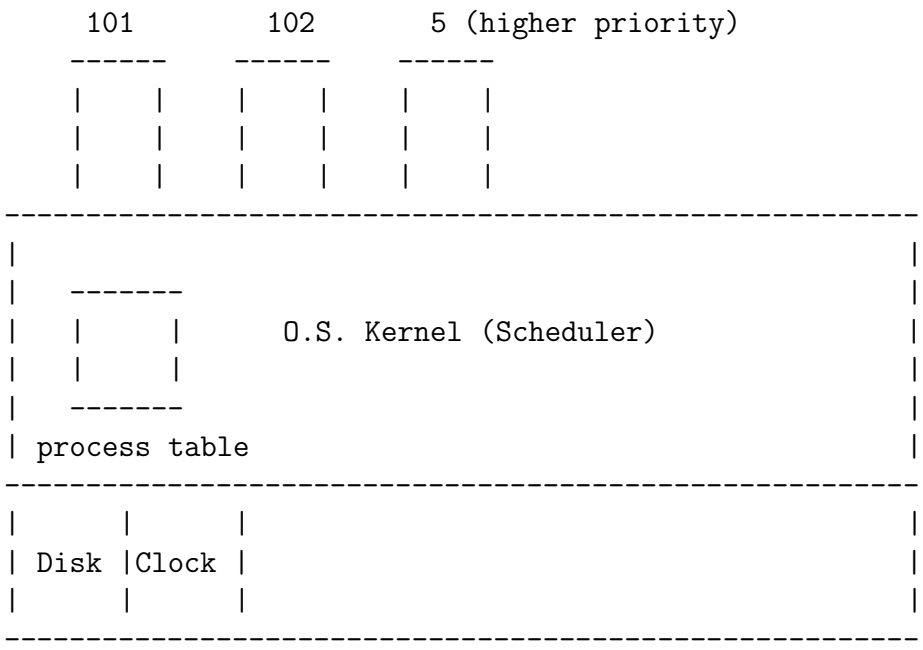
- program counter
- registers
- program status word
- stack pointer
- process state—blocked, ready, etc
- process priority
- status of open files
- other information (see Figure 2.4)

Process Id

The *process id* is used to identify a process. It may be a field of an entry or may be the index of a process in the process table.

Process States and Scheduling Example

Two user processes (pid 101 and 102). Monolithic kernel organization. A higher priority "server" process, such as a remote file server, waiting for requests.



Scenario

Event/Action	101	102	5
Initial State	Run	Ready	Block
Process reads input from keyboard	Block	Re/Run	Block
Process is humming along, request comes for file server	Block	Ready	Re/Run
File server handles request	Block	Re/Run	Block
Keyboard input is ready	Re/Run	Ready	Block
Clock interrupt, timeslice	Ready	Re/Run	Block
...			

Idle Routine

What to do if there are no processes eligible to run?

Execute code for a special "idle process" that executes an infinite loop. Whenever any process becomes eligible to run then switch to this process.

Could also got into reduced power mode on energy-constrained device.

Mutual Exclusion at the Lowest Levels

Disable interrupts.

Used to prevent unwanted interrupts. When? When modifying the process queues. For example, making a process *ready* or *unready* so that the process table is not left in an inconsistent state.