Lecture 3: Processes

Adam Hawley

January 23, 2019

1 Processes and Programs

- A **program** is an ordered set of specific operations for a computer to perform.
- A **process** is an abstraction of a running program.

A program is a passive entity stored on the disk including an executable file while a process is active and loaded into memory.

2 Process Management

Process management is one of the key functions of the OS, it includes:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Deciding which process should run
- Providing mechanisms for process synchronisation
- Providing mechanisms for process communication
- Provising mechanisms for deadlock handling

3 Process Structure

A process is more than the program code, which is sometimes known as the text section. It also includes the current activity such as the calue of the program counter and contents of the processor's registers. It also includes the process stack, which contains temporary data (such as function parameters, return addresses and local variables) and a data section which contains global variables. It may also include a heap, which is memory that is dynamically allocated during process run time.

See lecture slides for the process layout in memory. In particular, slide 9 on the **Process Control Block**

4 Process States

One thing stored in the PCB is the process state. Most OSs contain states which are similar or equivilent to the following:

- **Ready (or Runnable)** — scheduled to run but not currently using the processor
- **Running** — currently using the processor
- **Waiting (or Blocked)** — suspended waiting for some event to occur (such as a periodic clock tick or printer device to complete printing)
- **Terminated** — the process has finished execution (and will be removed from the system ASAP

See slide 11 for a useful flow diagram of states.

5 Process Lifecycle

- Processes are created
  - at system initialisation
  - by another process
  - by a user (e.g double clicking an icon)
  - batch job
- On creation, OS will:
  - allocate memory
  - create PCB
  - load program from disk to memory
  - copy arguments to program's stack and itialise registers

There are resource sharing options such as:

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

And execution options:

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

5.1 Ready and Running

When a process is created it is places into a ready queue. The scheduler decides which process to put into running state. The **policy** makes the decision itself and the **mechanism** uses the information in the PCB to return to the selected process.

5.2 Waiting/Blocked

Processes issuing I/O requests are placed in a waiting state. When the I/O completes, the process returns to the ready state. If all processes are waiting, then the processor is idle until one of the processes becomes unblocked and ready. This should be avoided at all costs.

5.3 Terminated

There are several different reasons a process can be moved into a terminated state:

- Normal exit (voluntary)
- Error exit (voluntary)
- Fatal error (involuntary)
- Killed by another process (involuntary)

5.4 Context Switch

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 via a context switch. Context of a process is represented in the PCB.

Context-switch time is pure overhead. The system does no useful work while switching. The more complex the OS and the PCB, the longer the context switch.

Switching is also dependent on hardware support. Some hardware proved multiple sets of registers per CPU which means that there can be multiple contexts loaded at once.

See slide 18 for a context switch diagram