Processes

These slides are created by Dr. Huang of George Mason University. Students registered in Dr. Huang’s courses at GMU can make a single machine readable copy and print a single copy of each slide for their own reference as long as the slide contains the copyright statement, and the GMU facilities are not used to produce the paper copies. Permission for any other use, either in machine-readable or printed form, must be obtained from the author in writing.

Process Concept

- A system executes more than one program at a time.
- Process – a program in execution.
  - Called task on Windows.
  - If you are running three copies of the same program, there will be three processes.
  - Try to open multiple Explorers and use Task Manager to see the number of tasks.
- Textbook uses the terms job and process interchangeably.
Process State

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

Diagram of Process State
The Existence of Processes in Memory

- PCB $i$
- PCB $j$
- Process Image $i$
- Process Image $j$

Kernel memory  Rest of memory

- Process Image: the process itself
  - Program code
  - Data
  - Shared libraries
- PCB: A kernel data structure that the OS use to keep track of a process
  - One PCB per process
Process Control Block (PCB)

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

<table>
<thead>
<tr>
<th>pointer</th>
<th>process state</th>
</tr>
</thead>
<tbody>
<tr>
<td>process number</td>
<td>program counter</td>
</tr>
<tr>
<td>registers</td>
<td>memory limits</td>
</tr>
<tr>
<td>list of open files</td>
<td>:</td>
</tr>
</tbody>
</table>

CPU Switch From Process to Process
Process Scheduling Queues

- 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.
- Process migration between the various queues.
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.
- Context-switch time is overhead; the system does no useful work while switching.
- Time dependent on hardware support.
### Process Creation

- 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.

### Process Creation (Cont.)

- **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.
Process Creation in Unix

```c
void main ()
{
    int pid;
    pid = fork();
    if (pid < 0) {error_msg}
    else if (pid == 0) { /* child process */
        execvp("/bin/ls", "ls", NULL);
    else { /* parent process */
        /* parent will wait for the child to
           complete */
        wait(NULL);
        exit(0);
    }
}
```

Process Termination

- Process executes last statement and asks the operating system to decide 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**).
  - Task assigned to child is no longer required.
  - Parent is exiting.
Cooperating Processes

- *Independent* process cannot affect or be affected by the execution of another process.
- *Cooperating* process can affect or be affected by the execution of another process.
- Advantages of process cooperation
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience

Inter-Process Communications

- Direct communication
  - Setup a channel between two processes
- Indirect communication
  - The sender stores the message in a mailbox, waiting for the retrieval by the receiving process.
  - The mailbox is a kernel data structure.
Synchronization

- Message passing may be either blocking or non-blocking.
- **Blocking**
  - Waiting for the operation to complete
  - considered *synchronous*
- **Non-blocking**
  - Proceeding without waiting for completion
  - considered *asynchronous*

Visualization
Why Multiple Identical Processes?

- Imagine you are creating a web server program
- Design Choice 1:
  - handle all clients with one process
  - The code will have to deal with thousands of concurrent users and thus quite complicated
- Design Choice 2:
  - Handle only one client
  - When seeing a new client, clone the process
  - Both the original and the clone handle only one client.

Remember that the cloning is done by the fork() system call.
Many servers are implemented this way.
There is a performance problem however.
  - Forking a process imposes significant overhead
  - The problem is worse when a server is handling thousands of clients (and thus processes)
We want to reduce the overheads of process creation/termination and memory footage.
Light Weight Process: Thread

- A thread uses the program code and data files of a hosting process.
- A thread has its own program counter, memory data and stack for independent execution.
- Reducing the overhead of multiple, identical processes.

Single-Thread Process

- Files in use
- Register Contents
- Program Counter

- Program Code
- Data and Stack
Multi-threaded Process

Register Contents  Program Counter  Files in use  Register Contents  Program Counter

Program Code

Data and Stack  Data and Stack