Lecture Notes on Dec/05

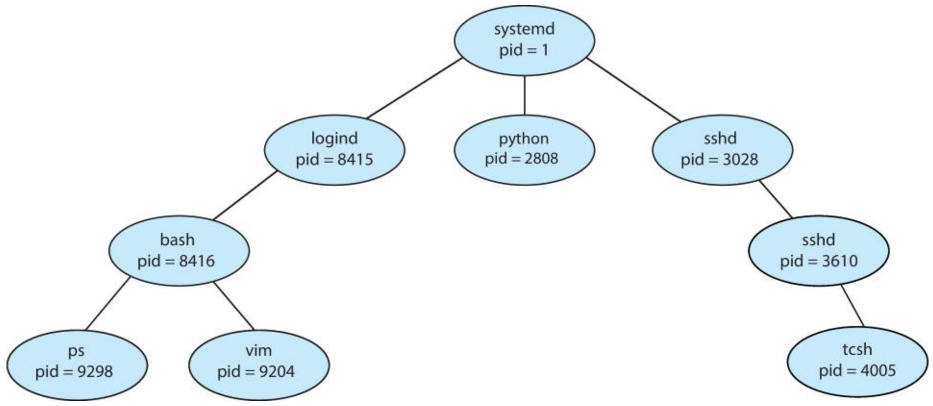
Inter-Process Communication – Part 2

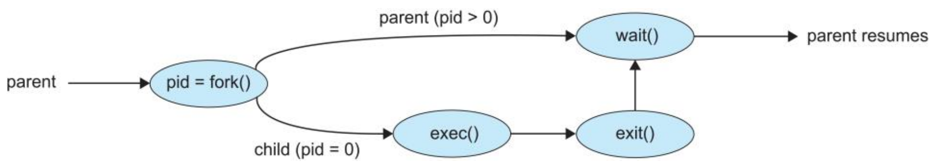
ECE217 Data Structure and Algorithms

Instructor: Dr. Shayan (Sean) Taheri

Process Creation

- **Parent** process create **children** processes, which, in turn create other processes, forming a **tree** of processes.
- Generally, process identified and managed via a process identifier (PID).
- **Resource Sharing Options**
  - Parent and children share all resources.
  - Children share subset of parent's resources.
  - Parent and child share no resources.
- **Execution Options**
  - 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.
  - Parent process calls **wait()** for the child to terminate.



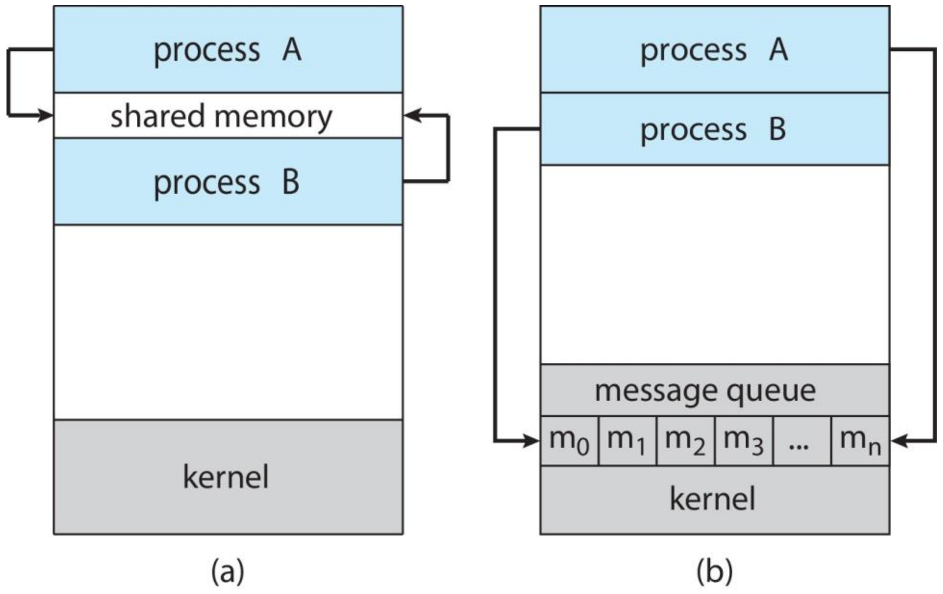


Process Termination

- Process executes last statement and then asks the operating system to delete it using the **exit()** system call.
  - Returns status data from child to parent (via **wait()**).
  - Process' resources are deallocated by operating system.
- Parent may terminate the execution of children processes using the **abort()** system call. Some reasons for doing so:
  - Child has exceeded allocated resources.
  - Task assigned to child is no longer required.
  - The parent is exiting and the operating systems does not allow a child to continue if its parent terminates.
- Some operating systems do not allow child to exists if its parent has terminated. If a process terminates, then all its children must also be terminated.
  - **Cascading Termination**. All children, grandchildren, etc. are terminated.
  - The termination is initiated by the operating system.
- The parent process may wait for termination of a child process by using the **wait()** system call. The call returns status information and the **PID** of the terminated process:
  - **PID = wait(&status)**;
- If no parent waiting (did not invoke **wait()**) process is a **zombie**.
- If parent terminated without invoking **wait** , process is an **orphan**.

Inter-process Communication (IPC)

- Processes within a system may be **independent** or **cooperating**.
- Cooperating process can affect or be affected by other processes, including sharing data.
- Reasons for cooperating processes:
  - Information Sharing.
  - Computation Speedup.
  - Modularity.
  - Convenience.
- Cooperating processes need **IPC**.
- **Two Models of IPC**:
  - **Shared Memory**.
  - **Message Passing**.
- **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.



- **Fig. (a): Shared memory.**
- **Fig. (b): Message passing.**

Inter-process Communication (Cont.)

- Paradigm for cooperating processes, **producer** process produces information that is consumed by a **consumer** process.
  - **unbounded-buffer** places no practical limit on the size of the buffer.
  - **bounded-buffer** assumes that there is a fixed buffer size.
- **IPC - Shared Memory**:
  - An area of memory shared among the processes that wish to communicate.
  - The communication is under the control of the users processes not the operating system.
  - Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.
- **IPC - Message Passing**:
  - Mechanism for processes to communicate and to synchronize their actions.
  - **Message System**: Processes communicate with each other without resorting to shared variables.
  - IPC facility provides two operations: **send(message)** and **receive(message)**.
  - The message size is either fixed or variable.
  - If two processes wish to communicate, they need to establish a **communication link** between themselves and exchange messages via send/receive instructions.

Assignment

- **Reading Assignment:**
  - Inter-process Communication Tutorial from **Tutorials Point**.
  - A Guide to Inter-process Communication in Linux from **Opensource.com**.

Questions?