Prelab 3

Andrew Todd

September 20, 2021

1. (a) **What's a Process?**
   A process is a sequence of executions regarded as a single entity by the OS for using system resources. In an OS kernel, each process is represented by a unique data structure.

   (b) **What's the meaning of pid/ppid**
   A pid is the current processes ID. The ppid is the ID of the parent process.

   (c) **What's the meaning of status?**
   The status is the current status of the process. It let's the system know when it's ready, free, etc.

   (d) **What's the meaning of priority?**
   Priority is how the system knows to schedule tasks. Some tasks are more important and have a higher priority.

   (e) **What's the meaning of event?**
   Event is the value that a process waits for while it's in the sleepList. When the event is triggered, the process can wake up.

   (f) **What's the meaning of exitCode?**
   The exit code let's the parent process know how the child exited so that it can act accordingly. If there is an issue, it can act given the exitCode.

2. (a) **What does fork do?**
   Fork creates a child task and puts it in the readyQueue.

   (b) **what does switch do?**
   When switch is called, it will save the current return address on the stack. Then it calls the scheduler, which decides which task to run next. When the execution returns from the scheduler, it restores the calling information.

   (c) **What does exit do?**
   First it will clear all of it's resources, then it will dispose of the child processes, if any. After that, it will throw an exitValue for its parent and become a zombie process. The parent will then check it and clear it.

   (d) **What does sleep do?**
   If it needs something that isn't there yet, the process will sleep to allow others to do their stuff. It will record an event value, change its status to sleep, enter a sleepList, then switch.

   (e) **What does wakeup do?**
   Wakeup will find the sleeping process in the sleepList and remove it. Then it will change the status to ready and put it in the readyQueue.

   (f) **What does wait do?**
   wait will wait for a zombie process, clear it, and add it back to the readyQueue.

3. **Test Fork**

   When fork was entered, it added process 2 to the readyQueue. Its parent is P1 and it was also removed from the freeList. P1 can only fork 7 times because there are a finite number of PROC structures.

4. **Test Sleep/Wakeup**

   P1 went into the readyQueue when I switched to P2.

   When running sleep, P2 goes into the sleepList with an event value.

   No PROC woke up with the new value because there aren't any PROC's in the sleepList with that value.

   When running wakeup with the same value I used for sleep, it woke up P2 because it matched P2's event.

5. **Test Child Exit / Parent Wait**

   When I run wait with P1, nothing happens because P1 does not have any children.

   After exiting P2, the parent process P1 is running. P2 is a zombie process.

   P1 cleared P2 and put it back into the freeList.

   P1 is put into the sleepList because it's waiting for the right event. P3 is now running.

   P3 exits and turns into a zombie, and since P1 was waiting already, it clears it up right away. P3 is clear and in the freeList.

6. **Test Orphans**

   P2 died, so it's children are passed onto P2's parent, which is P1.

   P2 is a zombie.

   P2 was added back to the freeList, and P2's former children stay with P1.

   After wait was run, P1 went into the sleepList, waiting for a dead process. To make P1 run again, you can wake it up by passing the correct value, or have a child die.