## Processes, Threads and Synchronization

Refer slide deck 04\_nto\_thread\_r11.pdf for answers.

### Processes

1. Briefly describe a process and its properties. What are some of the resources owned by a process? Can a process have priority?

A process own resources, and it cannot have a priority. A process owns its own memory space, file descriptors, timers, etc.

1. Briefly describe a thread. Can a thread have priority? What are some attributes of a thread? Explain what is a thread? Differentiate between a process and a thread.

A thread lives inside a process, and shares a process’s resources. It can have a priority, which determines when it can be executed within the current process.

1. A process has three threads. Does each thread have its own **errno** value?

Yes

1. A process has at least one thread. True/False.

true

1. Threads run code, processes own resources. True/False

true

1. Threads in a process share all the resources owned by a process. True/False

true

1. What is an advantage of a “Process Model”. (Slide 8)

The advantage of a process model is that you can add or remove different processes without affecting any of the existing ones. This isolation makes interaction that much more stable.

1. A specific scheduling algorithm is associated to a
   1. Thread \*
   2. Process
   3. Operating system
   4. CPU
   5. Critical resource
2. Each thread has its own stack. True/False  
   true
3. What are the three calls used to create a process?  
   fork, exec, spawn
4. What is the advantage in using spawn instead of fork and exec? (slide 21)
   1. spawn provides more control. It is efficient compared to fork and exec.
   2. \*spawn can be used in a multithreaded process.
   3. spawn does not copy the data segment. (Data segment is the address space of a program that is used to store global and static variables.)

1. How does the parent know when a child process terminates? When does a child process become a zombie? How can a zombie be prevented?  
   sigchild(). Becomes a zombie when the parent loses track of it. Zombie can be prevented by waiting for it to close before exiting.

1. QNX does not support **fork()** in a multithreaded process. True/False  
   true
2. Fork creates an identical copy of the process. At which point in the code, does the new created process, start executing? Hint: Is it from the start of the process? (slide 17)

Will start from the fork() call

1. Name some resources that are inherited when a new process is created. Name two resources that are not inherited. (slide 18)

Memory, file descriptors

Side channels, channels, timers

1. Briefly explain the **exec()** function. What are some inheritance rules in exec()? Does the process id (pid) remain the same after calling exec()?  
   exec replaces the current executing environment with a new one. Inheritance rules are the same as with fork, except the address space is not copied. PID stays the same.
2. What is the advantage of spawn() compared to exec() and fork()?  
   will load and run a program in a new process will return the pid of the child process inheritance rules follow that of fork() and exec\*()
3. Name the signal received by the parent process when the child process exits.  
   SIGCHLD
4. Name the function that provides the reason to the parent process when a child dies. (slide 24)  
   waitpid() or wait()
5. Write the line of code that prevents the notification to the parent when a child dies.

1. What is the state of the child process if the parent does not call **wait()**?

### Threads

1. In the following line of code

**pthread\_create (&tid, &attr, &func, &arg);**

explain the purpose of

&tid, thread id – **Identifies the pthread structure to create**

&attr, thread attributes, such as priority, scheduling algorithm

**Determine how the thread will behave, when it will run**

&func the code to execute

&arg, arguments passed to the function

1. What is the default priority of the thread and the default scheduling algorithm?

Default priority : 10

Default scheduling algorithm: Round Robin

1. True/False. Each thread has its own stack.

true

1. When does a thread execute after it is created?
   1. it runs immediately
   2. it runs after the main thread has completed execution
   3. **\* it is placed in the ready queue, it will run when it has its turn for CPU**
   4. the programmer must explicitly give a command for it to run
2. Exercise: Review the sample code in example\_thread.c from QNX. Compile and run the sample code. Write a function, call it functionC. Create another thread in the code, execute functionC. Print its thread\_id.
3. True/False. In a process you cannot create two threads with exactly the same code.

False

1. A program is running and you press Ctrl-C at the console
   1. The program goes to sleep
   2. **A SIGTERM signal is sent to the process and the process terminates**
   3. A SIGINT signal is sent to the process and the process terminates
   4. The program is taken off from the ready queue
2. True/ False. Only the main thread can check the status of another thread. No other thread can check its status. (slide 35)  
   false
3. What does the sched\_yield() function do? (check Momentics Help)  
   yields to other threads of the same priority.
4. The function atomic\_add\_value shown below increments a variable by a specified value. What additional feature does the function offer?  
   unsigned atomic\_add\_value( volatile unsigned \* loc, unsigned incr ); (check Momentics Help)

Thread safety

1. Consider the scenario. Thread **t1**, at priority 10, calls **pthread\_mutex\_lock()** to lock a mutex. **t1** successfully locks the mutex so the call returns. A little later, thread **t2**, running at priority **12**, tries to lock the same mutex. Since t1 has it locked, **t2** will block on the **pthread\_mutex\_lock()** call. Explain what happens to the priority of thread t1. Give a reason for your answer. (refer slide 40).

T1 is bumped to the priority of t2 until the mutex is unlocked to prevent t2 from hogging the cpu while it is waiting for t1 to unlock its resource.

1. What does the function pthread\_join() do? When would you use this function?

Waits for the specified thread to finish execution and returns its exit status.

### Mutex

1. What is a mutex? Suggest an example where you would use a mutex.

Mutual exclusion to a resource. You would use a mutex to protect a file from corruption during write.

1. Can a mutex be shared between processes? How? (slide 59)

Set PTHREAD\_PROCESS\_SHARED when creating the mutex

1. How does a mutex guarantee its functionality? In what circumstance will a mutex fail to function?

Guarantees functionality by locking the mutex and preventing other threads from accessing it. It will fail if other threads bypass the mutex altogether.

1. What does the following function do? **pthread\_mutex\_lock( &var\_mutex );**

Locks the mutex var\_mutex

1. How is the function **pthread\_mutex\_trylock** different from **pthread\_mutex\_lock** ?

Pthread\_mutex\_trylock will simply test the mutex to see if it is locked, and not actually lock it. Pthread\_mutex\_lock will block until it can lock.