Jan 25, 2005
Lecture 3: Inter Process Communication

February 17, 2005

1 ADMIN

READING FOR THIS WEEK: Please start reading Chapter 2.

Homework 1: Please pickup from website. Due on Feb 1.

2 Review

"Process Management"

Q: Name the possible proces states

A: Running, Ready, Blocked.

Q: Draw the graph of possible state transitions (be sure to give the proper name to each transition, and do not forget the transition in which processes are created or destroyed).

A:

- preempt or timeout: running → ready
- run: ready → running
- event-wait: running → blocked.
- event-occurred: blocked → ready.

THE 7-STATE PROCESS MODEL: We can elaborate the above model in two ways:

First, we can add two "implicit states": New, Exited. All processes are assumed to be originally in the New State, and eventually end up in Exited State.

The new transitions are:

- create or fork: new → ready.
- exit or terminate: running → exited.

- killed: blocked → exited.

We can further introduce the concept of a "suspended process". Suspended processes are those whose core image have been swapped out into disk. This is important to support a large number of processes, and also for efficiency.

Suspended processes can be in one of 2 States: **suspended-ready** or **suspended-blocked**

Additional TRANSITIONS:

- suspended/swapped-out: blocked → suspended-blocked.
- activate/swapped-in: suspended-blocked → blocked.
- event-occured: suspended-blocked → suspended-ready.
- suspended/swapped-out: ready → suspended-ready.
- activate/swapped-in: suspended-ready → ready.

3 Homework 1 and Cygwin Environment

Go over my Cygwin FAQs (from class website)

go over the use of "make" program: structure of a Makefile

go over homework

4 Fork and Exec

Q: How does the unix shell use forking in its basic loop?

A: [p.48,49] It reads a command, forks a child to execute the command, and then waits for the child to finish.

Q: How is the child process and parent process different immediately after a fork? Give a simple program statement in the program to use this difference.

A: The child process gets the value 0 from the fork, the parent process gets the PID from the fork. So the typical program fragment is:

```
if (fork() != 0) {
... do the parent action...
}else{
... do the child action...
}
```

Q: What are typical parent/child actions in the above code?

A: Parent Action: wait for child to complete (e.g. the shell fork) Child Action: call "exec" to execute some command. There are several versions of "exec" (execv, execve, execle).

Q: Why is it said that fork and exec command goes hand-in-hand?

A: Either one alone is pretty limited in use.

[Question for thought: What scenarios are there where you would use one without the other?]

There are 6 VARIANTS of exec command: the first argument of each of variant is the name of the program to execute. E.g., execl("ls", ...) says that we want to execute the unix "ls" program. The remaining arguments depends on the variant AND on the program to be executed:

System Call	Argument Format	Environment Passing	Path Search
execl	list	auto	no
execv	array	auto	no
execle	list	manual	no
execve	array	manual	no
execlp	list	auto	yes
execvp	array	auto	yes

The arguments to the program to be executed can be given in ARRAY format or in LIST format:

E.g., execl("ls", ".", NULL);

In this example of the list format, we see that the list of arguments is terminated by a NULL argument.

The environment for executing the program can also be passed explicitly or implicitly inherited from the current environment.