15-395 Lab 5
Process Tools

Objective

This lab is designed to give you a perspective on the resources associated with the process and one of the ways the kernel exports that information. It is also designed to give you a chance to get back in the groove of writing C.

Times of Interest

Assigned: Wednesday, September 3, 2008
Due: Tuesday, September 9, 2008

ps: Process Status

Please play around a bit with the using tool *ps.* It is the standard way to get information about running processes. This assignment asks you to implement a very similar tool. Before beginning this part of the assignment, please "*man ps"* and give it a try.

As with *ps*, the output should be nicely formatted. And include a header, unless otherwise specificed on the command-line. Given a *pid* it should produce a single line summary of the process's vital statistics:

- command name, e.g., argv[0]
- state, e.g., S, R, T, Z, W, etc
- pid
- ppid
- uid
- pgrp

The following flags should be supported:

- no flag, reports on all processes using the same terminal as the *ps*, itself
- -a, reports on all processed on the system
- -u *uid*, reports on all processes owned by user with *uid*
- -u *username*, reports on all processes owned by *userid*
- -p *pid*, report on only the process with pid *pid*
- -pp *ppid* report all processes with *ppid* as their parent.
- -n, omit the header

Resources

You might want to do a “man 3 getopt” and a “man opendir”.

Common Sense

If your project contains code which requires assembly (compiling, generation, 7c), it must include a *Makefile* that can build either of, or all of, the tools in your suite. This file should be well-written, represent all of the dependencies, and should include a *clean-rule*.