

Introduction to Processes on Unix –tester1 and tester2 can be found on web page

Students, this lab will take longer than the tutorial period. Don't expect to just sit at a machine and the answers will come to you. You **MUST** read the man pages and proc files etc to answer the questions. Two shell windows (one on the man pages and one for the /proc files is a good idea as well as a browser ready to look up definitions). The whole point of the exercise is the following:

- You become more familiar with Unix
- You get the hang of reading help on Unix to LEARN new stuff
- You back up the theory in the Processes lectures

Using a LOCAL shell windows only, answer the following questions:

1. Using the 'more' or 'less' file-browsing commands **Browse** the files in the /proc directory and answer the following:
 - a. What is /proc? ____
 - b. What is an IRQ? ____
 1. List the IRQ and number of interrupts for the timer and keyboard?
 - c. Where can you find the machine's up-time? ____
 1. What is the up-time of the machine mean?
 2. What is a jiffy? _____
 3. What is the up-time for this machine in jiffies? ____
 4. Look at your neighbours, different? Why? ____
 - d. What version of the OS is running? ____
 - e. What does it mean by 'load average'? ____
 1. What is the current load average of your machine over the last minute? ____
 - f. Look at the PCI, what does it mean?
 1. How many devices are there? ____
 2. How many USB slots? ____
 3. What is the IRQ number of your Ethernet controller? ____
 - g. What are mounts? ____.
 - h. Look at memory information, what is the total memory ___, free memory ___ and cache allocation? ____
 1. Look at the modules loaded, how many? ____ what is NFS? ____
2. Run the '**top**' command in one window and observe its output.
 - a. Look at its man page and study the different fields listed by 'top'
3. Download the tester1.c program.
It is a process that runs for 10 seconds.
 - a. Compile and convert it into a process called *tester1*. (gcc -o tester1 tester1.c then ./tester1 ____ you might have to run it a couple or times)

b. Observe the output of the 'top' screen.

c. Tell me what:

1. Its process ID is _____

2. Does it have a parent? _____

3. What do you think a parent is? _____

4. What is the size of the task? _____

5. How much of this is code? How much is data/stack? _____

6. How much memory is this task using? _____

7. Why is it so low? _____

8. How much CPU is consumed by the task? _____

9. Now look at what state the task is in, is it sleeping, running, stopped? _____

4. Now stop this process by a 'ctrl-z' and see how the change is reflected in the top.

a. What state is registered and what memory is being used and what CPU is being used for that task. _____

b. How does this affect the overall statistics for all processes? _____

c. That is, what is the utilisation of all the machines resources compared with when the program was running? _____

5. Bring the process to running state again by executing command 'fg'. See the change in 'top' screen. Write these changes down.

a. This tester1.c process will automatically exit after a certain timeout. _____

b. You may need to rerun it to complete the 'experiment'. _____

6. Now make the alarm size much larger.

a. Make the program into a process (as before). _____

b. How does this affect the performance of the task and the overall performance? _____

7. Look at the top process's own statistics.

a. What can you tell me about top's resource consumption? _____

b. Why do you think it is has these statistics? _____

c. What if it didn't? _____

8. Run the 'ps' command.

a. Look at its man page.

b. What is the difference between ps and top? _____

a. Explore various parameters to ps and study the output. How do you look at your processes only? _____

- b. Conduct the same experiment as above; observe the state of your process in the output reported by 'ps'. _____

9. Download tester2.c,

Now look at the overall process statistics, what's happening in *this* program? Tell me what:

- a. Its process ID is _____
b. What is the size of the task? _____
c. How much of this is code? How much is data/stack? _____/_____
d. How much memory is this task using? _____
e. How much CPU is consumed by the task? _____
f. Why is this different from tester1? _____
g. Now look at what state the task is in, is it sleeping, running, stopped? _____

10. **How does ps, top and proc relate to each other?**