THE UNIVERSITY of NEW ORLEANS

DEPARTMENT OF COMPUTER SCIENCE

CSCI 2467, Spring 2020

Class Activity: file descriptors and the /proc filesystem

1 Files and file descriptors

We saw in class that `open()` returns a small integer called a *file descriptor* which is then used by a process when calling `read()`, `write()`, `close()` and other file related functions. Our purpose today is to observe how this works with real programs running on `systems-lab`.

1.1 Preparation

Attention students: this activity will only work fully from the HP terminals in rooms 209 and 212. Please log in to the lab rather than connecting via ssh.

To demonstrate file descriptors, we want to use a PDF file as an example. We can use the slides from Chapter 10 for this, which is available on the course website.

1. Start by logging in and opening a terminal.
2. We can fetch the PDF from the command line using the `wget` command:
   `$ wget http://2467.cs.uno.edu/lectures/08io.pdf`
3. view PDF using `evince`: `$ evince 08io.pdf`

   Now you should see the slides on your screen. (Evince is a program for viewing documents like PDF files.)

2 The `/proc` filesystem

UNIX systems have a special filesystem for information about running processes [1] called `/proc`. The Linux kernel represents all currently running processes with a directory of the form `/proc/PID` (where PID is the integer process ID).

2.1 Find evince in `/proc`

1. While the PDF viewer is open, start another terminal window by using the menu on the existing terminal to do: **File / Open Terminal / 0. Default**
2. In the new terminal, find your evince process using: `ps -u $USER`
3. What is the PID of evince? (not evinced) Write it here: `_____`

   (If you have trouble finding evince in the ps output, try using: `pgrep -u $USER -x evince`
4. Change to that directory with: `$ cd /proc/PID`

[1] Killian, T. J. (1984). Processes as files. In *USENIX Association Conference Proceedings (pp. 203-207).*

5. There should be a directory under that PID called `fd`.
   Examine it with: `$ ls -l fd`

   Do you see 08io.pdf? What number file descriptor is it? `_____`

6. On the left side of the `ls -l fd` output we see the letter l followed by some letters and dashes. These are the file mode (permission) bits. (We looked at this briefly in intro lab, please refer back to that if necessary.) What permissions are set for the file descriptor for 08io.pdf? What does this tell us about how the file was opened?

7. There is another interesting file for each processes called `status`. Examine this file, either using the `cat` command or by using a text editor such as `nano`.

8. Look at the fields: State, PPid, SigPnd, SigBlk. What do these mean?

9. Pay attention to the final two lines: `voluntary_ctxt_switches` and `nonvoluntary_ctxt_switches`. Click on the PDF, change pages, and interact with evince. Then re-open `status` and see how these values changed. See if these change when you do not interact with evince.

10. Go back to your first terminal (there may be some warnings printed by evince on here which we do not pay attention to). Type CTRL-Z. This will send SIGTSTP to evince. Re-read the `status` file for evince - what has changed?

11. Try to change pages or interact with evince. What is happening now?

12. Put evince in the background on the first terminal by typing `bg %1` (bash has a `bg` built-in command very similar to the one you made in `tsh`).

13. Now can you interact with evince? How has its status changed? How about the `voluntary_ctxt_switches` ?

14. Close evince. Now what happens when you type `ls` in the terminal in the `/proc` filesystem?

2.2 Examine `/proc` when using input/output redirection

1. Change to your shell lab working directory. Run the program `./myspin 300`. Find its PID using `ps` as before.

2. In the other terminal, examine the `/proc/PID/fd` directory for that myspin process using `ls -l`. What file descriptors are open? What does this mean about the current myspin program?

3. After this, terminate myspin using CTRL-C. Run it again, but with input redirection: `$ ./myspin 300 < README`

4. Now examine again `/proc/PID/fd` for the new PID - what has changed?

5. Do this again with output redirection: `$ ./myspin 300 > output`

6. Examine `/proc/PID/fd` again and note the differences.