Lab11\_Project3\_Milestone3\_Report

Yukui Ye

SUID:439644268

There are three mode:1 print in parent;2 print in children;3 print into file

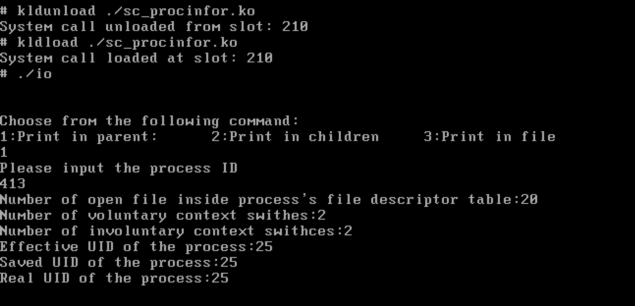
**Code logical flow:**

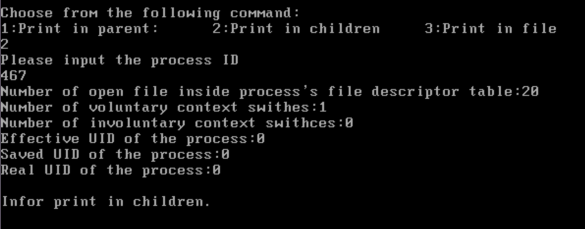
First create five pipe, you can check the code to see the using property of each of them, After we fork, There are two process running at the same time, which are parent and children process. In the parent’s for loop, set up select environment since ever time you need to manage the set of descriptor, if 0 descriptor is true, then go to scanf two variable from user-input pid and choice, and write them specifically to piple2(fd2) and pipe3(fd3), then it will goes to children process, and parent process is waiting for children process’s return value.

In children process, first read in choice send from parent through pipe3, and separately into three different mode, if choice is 1, the system call information should be printed in parent mode, it first create a buffer to store all the information received by syscall, and then write the buffer to pipe1(fd1), record the choice to pipe4(fd4) at the same time so that parent would read from pipe4 to know which choice is returned from children, then the parent would goes to its seconde if sentence since fd1[0]is true now order to print out relatively information. Meanwhile, due to different return format based on whether you can find the process pid or not, you need to write (yes) ‘y’or (no) ‘n’ to pipe5(fd5), then parent would read pipe5 to know which format to print out. The three mode is working similar like above explanation. When you do print in children mode(2),it goes to print out the process information directly and send “print in chidren”message to parent. While when you do print in file, it first record all information to an array, and then create a file named ‘mollysmile\_ye’, and write the array into the file. Alternate choice , you can also create a file named “molly.txt” to copy the process information by dupe standout 1 descriptor. Detail information, please look at the code file’iosyscall.c’.

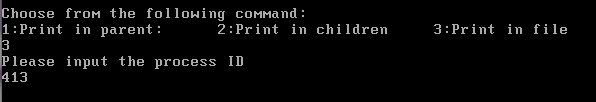
**Comparision between different mode**

The following first screenshot is the result for different printout mode. First example, is when you type ‘1’, it goes to print out in parent mode, input valid process pid’413’, it prints out information through parent. Second example illustrates how it works when you type’2’ which goes to print out in children mode, type valid process pid, it prints out directly through children side to terminal, and it return message ‘print in children’ to parent side, and parent print out this message to terminal. If you choose ‘3’, and type process pid ’413’, it prints out information to file named’mollysmile\_ye’, go to check the file, it shows the information of process with pid413.

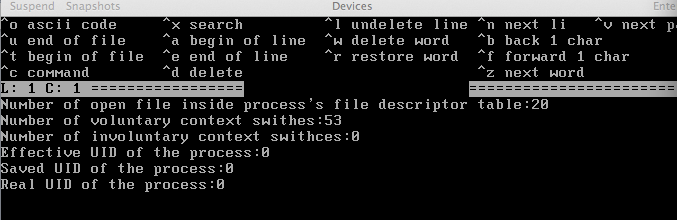
****

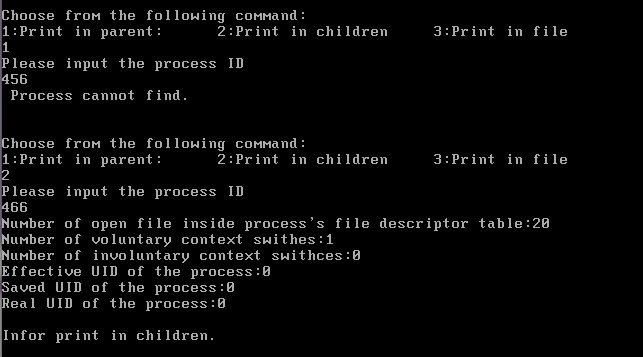
****

**choose to print in file**



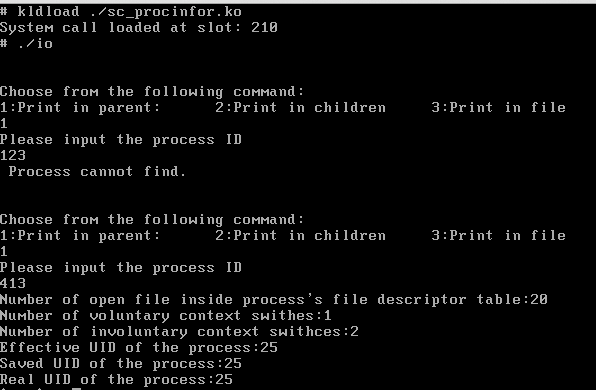
**go to file check printout information**



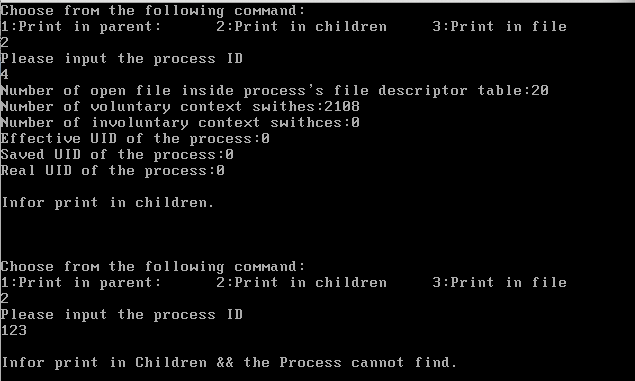
****

**Comparision within one mode based on PID can find or cannot find**

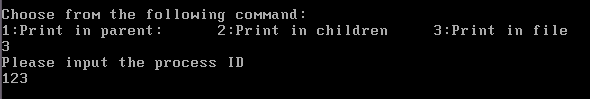
1)Print in parent mode1: first type an unexist process pid123,it first recodes whether the pid is exit or not through pipe5(fd5),then prints out “process cannot find ”information through parent process code; when we type an exit process pid 413, it prints out all the process information through parent side to terminal.



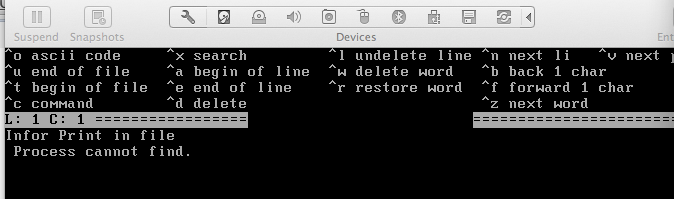
2)Print in children mode: First type’2’to choose this mode, then type a valid process pid, it prints out process information directly through children side to terminal, and send the message ’Infor print int children’ to parent side, then parent print this message to terminal. If the pid is invalide, it sends ‘Infor print in Children & the process cannot find ’, then parent print out this message to terminal.

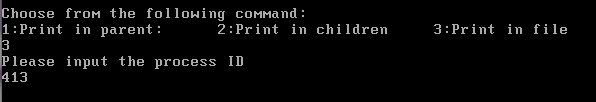


3)Print into file: type’3’,goes to print in file mode, type invalid process pid, go to check the relative file , find the information ‘Infor print in file process cannot find’; do another example, input valide pid’413’, and go to check the file, it prints out the relative information of this process in the file.



check the file:





check the file:

