CS 451 Operating Systems

Writing Your Own PSTREE

ASSIGNMENT 1

All unix-like systems provide a system program named `pstree`. This command outputs the the process hierarchy of all the processes in the system in a tree-like format. When you type `pstree -p` you will see output similar to the one below. Note that process ids (PIDS) are included in the `-p` option.

```
systemd(1)─┬─VBoxService(955)─┬─{VBoxService}(956)
           │                  ├─{VBoxService}(957)
           │                  ├─{VBoxService}(958)
           │                  ├─{VBoxService}(959)
           │                  ├─{VBoxService}(960)
           │                  ├─{VBoxService}(961)
           │                  └─{VBoxService}(962)
...
```

The above output means that process 1's child is process 955 and process 955's children are 956, 957, etc.

In this assignment you will write a variation of this original `pstree` command called `mypstree` with following changes

a. `mypstree` command will take only one option which is a `pid` indicating where the tree should begin. If the `pid` provided does not corresponding to an existing process, the program prints an error message and quits
b. `mypstree` will only list `pids` and not the name of the processes as opposed to original `pstree`
c. `mypstree` will not include any threads. The original `pstree` includes threads as well. Therefore you will see much less output from `mypstree`
d. `mypstree` output will be similar to the original pstree's output but not exactly the same due to the reasons above.

When your program is run using following syntax,

```
mypstree 1
```

it should display an output as follows:

```
Children of 1: 955
         Children of 955: 956 957 958 959 960 961 962
                  Children of 956:
                  Children of 957:
                  Children of 958: 2022
                  Children of 959:
                  Children of 960:
                  Children of 961:
                  Children of 962:
                           Children of 2022:2023
```

etc.

The output need not be sorted. The output may not be aligned perfectly due to some process IDs occupying one digit, some two digits and some three digits etc. However, each process's children must be listed correctly and the entire tree must be traversed correctly. Even if a process has no children, it must be listed with no entries after the ":".

Note also that the print is in level order of the tree, i.e, each level is indented to the right of the previous level.

The information you need to generate the tree above is located in `proc` file system, under `/proc` directory. You must read the man page of `proc` file system in order get familiarity of `proc` file system.

The `/proc` directory contains one directory entry for each currently running process. The children of each process is stored under that directory. To be precise, the children of each process is available at `/proc/pid/task/pid/children`. For example, the children of process 1 is at `/proc/1/task/1/children`. You must now travel the child directories using `/proc/childpid/task/childpid/children` to find the children of `childpid`.

There may be some thread ids listed `/proc/pid/task` directory. You should not travel these thread ids but only process ids.

Getting Started

You will need understanding of two main concepts in completing this assignment, beyond the knowledge basic C syntax.

1) Knowledge of `proc` file system and output of `pstree` which you can obtain by reading man `proc` pages and running `pstree`.
2) Traveling the tree of processes in level-order. The outline of code for level-order traversal is given below. We will use a Queue that is first in first out. Note that the code is not recursive and it is not expected to be recursive.

```
enque root to a Queue Q (to the end of the queue)
 while (Q is not empty)
     Node n = deque(Q) (from the front of the queue)
```

```
        for each child c of the node n:
            enque the node c to the Q;
        endFor
    endWhile
```

If you need to recognize when you are moving from level to level, there are several techniques: Here is a “marker” technique:

```
enque root to a Queue Q (at the end of the queue)
enque a marker to the Q (marker could be -1)
while (Q is not empty)
    Node n = deque(Q); (from the front of the queue)
    if n is the marker
       Increase level number by 1
       enque a marker to queue Q;
       n = deque(Q);
    endIf

    for each child c of the node n:
         enque the node c to the Q;
    endFor
endWhile
```

Code Development

Code development must be done in stages. Each time as new functionality is added, you must be checking to make sure it works. For example, you could start with reading command line option and successfully reading /proc/pid/task/pid/children. Then move to traversal. Writing entire code in one go and debugging would lead to frustration and you will spend more time on the assignment.

You can use arrays instead of the queue data structure. As long as you are processing elements from one end of the array and adding to the other end of the array, it will function as a queue.

If you use malloc, you must free the memory after use. Otherwise you will encounter segmentation faults.

To check your code thoroughly, run

```
mypstree 1
```

and

```
mypstree 2
```

These two are initial processes. The number of lines of output when totaled, must match with number of process listed in

```
ps –aef
```

To find number of lines in a file, use the "`wc`" command.

To redirect the output of your program in a file named "`out.txt`", type

```
mypstree 1 > out.txt
```

Style and Comments

Follow the style guide given on Blackboard. Make sure all of your variables are meaningful and code is indented appropriately for ease of reading.

Error Checking

No errors are expected in the input. The command will contain exactly one number indicating the `pid`. If that `pid` does not exist you should print an error message. No other error checking is expected.

What should I turn in

Turn in your C file, README file and DEMO file as described in lab policies document on Blackboard. In doing the DEMO file, make sure you run the program with various pids, particularly with pid 1, pid 2 and one other pid.

Questions?

Ask often and ask early…