Homework #1

Advanced Programming in the UNIX Environment

Due: Apr 7, 2021

Read carefully before you implementing and submitting your homework.

Implement a 'lsof'-like program

In this homework, you have to implement the 'lsof' tool by yourself. 'lsof' is a tool to list open files. It can be used to list all the files opened by processes running in the system. The output of your homework is required to *strictly* follow the spec, the TAs will use the 'diff' tool to compare your output directly against our prepared sample sample testdata. Spaces and tabs are compressed into a single character when comparing the outputs.

A sample output from this homework is demonstrated as follows:

```
$ ./hw1  -c bash
bash
COMMAND PID     USER            FD      TYPE     NODE     NAME
bash    26884   terrynini38514  cwd     DIR      57779    /media/psf/Home/Desktop
bash    26884   terrynini38514  root    DIR      2        /
bash    26884   terrynini38514  exe     REG      1179741  /usr/bin/bash
bash    26884   terrynini38514  mem     REG      1179741  /usr/bin/bash
bash    26884   terrynini38514  mem     REG      1186555  /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
bash    26884   terrynini38514  mem     REG      1185120  /usr/lib/locale/locale-archive
bash    26884   terrynini38514  mem     REG      1185791  /usr/lib/x86_64-linux-gnu/libc-2.31.so
bash    26884   terrynini38514  mem     REG      1185926  /usr/lib/x86_64-linux-gnu/libdl-2.31.so
bash    26884   terrynini38514  mem     REG      1186902  /usr/lib/x86_64-linux-gnu/libtinfo.so.6.2
bash    26884   terrynini38514  mem     REG      1708797  /usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
bash    26884   terrynini38514  mem     REG      1185576  /usr/lib/x86_64-linux-gnu/ld-2.31.so
bash    26884   terrynini38514  0u      CHR      3        /dev/pts/0
bash    26884   terrynini38514  1u      CHR      3        /dev/pts/0
bash    26884   terrynini38514  2u      CHR      3        /dev/pts/0
bash    26884   terrynini38514  255u    CHR      3        /dev/pts/0
```

The detailed spec of this homework is introduced as follows. Your program has to output the following fields (columns) for each file opened by a running process. Each line presents the information for a single file. The required fields include `COMMAND`, `PID`, `USERM`, `FD`, `TYPE`, `NODE`, and `NAME`. The meaning of each field (column) is introduced below.

- `COMMAND`:
  - The **executable filename** of a running process.
  - DO NOT show arguments.
- `PID`:
  - Process id of a running process.
  - Only need to handle opened files in process level (check `/proc/[pid]`. No need to handle opened files in thread level (that would be in `/proc/[pid]/task/[tid]`).
- `USER`:
  - The username who run the process.
  - Please show `username` instead of UID.
- `FD`: The file descriptor. The value shown in `FD` field can be one of the following cases.
  - `cwd`: The current working directory, can be read from `/proc/[pid]/cwd`.
  - `root`: root directory, can be read from `/proc/[pid]/root`.
  - `exe`: program file of this process, can be read from `/proc/[pid]/exe`.
  - `mem`: memory mapping information, can be read from `/proc/[pid]/maps`.
  - `del`: indicate that the file or link has been deleted. You should show this value if there is a (deleted) mark right after the filename in memory maps.
  - `[0-9]+[rwu]`: file descriptor and opened mode.
    - The numbers show the file descriptor number of the opened file.
    - The mode "r" means the file is opened for reading.
    - The mode "w" means the file is opened for writing.
    - The mode "u" means the file is opened for reading and writing.
  - `NOFD`: if `/proc/[pid]/fd` is not accessible. In this case, the values for `TYPE` and `NODE` field can be left empty.
- `TYPE`: The type of the opened file. The value shown in `TYPE` can be one of the following cases.
  - `DIR`: a directory. `cwd` and `root` is also classified as this type.
  - `REG`: a regular file
  - `CHR`: a character special file, for example

    ```
    crw-rw-rw- 1 root root 1, 3 Mar 17 17:31 /dev/null
    ```
  - `FIFO`: a pipe, for examle
    - A link to a pipe, e.g.,

      ```
      lr-x------ 1 terrynini38514 terrynini38514 64 Mar 17 19:55 5 -> 'pipe:[138394]'
      ```
    - A file with `p` type (FIFO)

      ```
      prw------- 1 root root 0 Mar 17 19:54 /run/systemd/inhibit/11.ref
      ```
  - `SOCK`: a socket, for example

    ```
    lrwx------ 1 terrynini38514 terrynini38514 64 Mar 17 19:55 1 -> 'socket:[136975]'
    ```
  - `unknown`: Any other unlisted types. Alternatively, if a file has been deleted or is not accessible (e.g., permission denied), this column can show `unknown`.
- `NODE`:
  - The i-node number of the file
  - It can be blank or empty if and only if `/proc/[pid]/fd` is not accessible.
- `NAME`:
  - Show the opened filename if it is a typical file or directory.
  - Show `pipe:[node number]` if it is a symbolic file to a pipe, e.g.,

    ```
    l-wx------ 1 ta ta 64 三 8 02:11 91 -> 'pipe:[2669735]'
    ```
  - Show `socket:[node number]` if it is a symbolic file to a socket, e.g.,

    ```
    lrwx------ 1 ta ta 64 三 8 02:11 51 -> 'socket:[2669792]'
    ```
  - Append ` (deleted)` (note the space before the message) to the end of the value if the value for `FD` is `del`.
  - Append ` (opendir: Permission denied)` if the access to `/proc/pid/fd` is failed due to permission denied.
  - Append ` (readlink: Permission denied)` if the access to `/proc/pid/(cwd|root|exe)` is failed due to permission denied.

Program Arguments

Your program should work without any arguments. In the meantime, your program have to properly handle the following arguments:

- `-c REGEX`: a regular expression (REGEX) filter for filtering command line. For example `-c sh` would match `bash`, `zsh`, and `share`.
- `-t TYPE`: a TYPE filter. Valid TYPE includes `REG`, `CHR`, `DIR`, `FIFO`, `SOCK`, and `unknown`. TYPEs other than the listed should be considered as invalid. For invalid types, your program have to print out an error message `Invalid TYPE option.` in a single line and terminate your program.
- `-f REGEX`: a regular expression (REGEX) filter for filtering filenames.

Homework Submission

We will compile your homework by simply typing 'make' in your homework directory. You have to ensure your Makefile produces the executable hw1. Please make sure your Makefile works and the output executable name is correct before submitting your homework.

Please pack your C/C++ code and Makefile into a **zip** archive. The directory structure should following the following illustration. The *id* is your student id. Please note that you don't need to enclose your id with the braces.

```
{id}_hw1.zip
└── {id}_hw1/
        ├── Makefile
        ├── hw1.cpp
        └── (any other c/c++ files if needed)
```

Your have to submit your homework via the E3 system. Scores will be graded based on the completeness of your implementation.

Remarks

- Please implement your homework in C or C++.
- Using any non-standard libraries and any external binaries (e.g., via system()) are not allowed.
- No copycats. Please do not use codes from other (even open sources projects).
- We will test your program in **Ubuntu 20.04 LTS** Linux with the default gcc version (9.3.0).

Outputs

Your program has to order the output lines by performing a numeric sort against process ID (PIDs). We will test your program with and without root permission. If an operation does not have sufficient permission to perform, you have to print out `Permission denied` message.

Run the command without root permission

```
$ ./hw1 | head -n 20
COMMAND                 PID       USER      FD        TYPE      NODE      NAME
systemd                 1         root      cwd       unknown             /proc/1/cwd (readlink: Permission denied)
systemd                 1         root      root      unknown             /proc/1/root (readlink: Permission denied)
systemd                 1         root      exe       unknown             /proc/1/exe (readlink: Permission denied)
systemd                 1         root      NOFD                          /proc/1/fd (opendir: Permission denied)
kthreadd                2         root      cwd       unknown             /proc/2/cwd (readlink: Permission denied)
kthreadd                2         root      root      unknown             /proc/2/root (readlink: Permission denied)
kthreadd                2         root      exe       unknown             /proc/2/exe (readlink: Permission denied)
kthreadd                2         root      NOFD                          /proc/2/fd (opendir: Permission denied)
rcu_gp                  3         root      cwd       unknown             /proc/3/cwd (readlink: Permission denied)
rcu_gp                  3         root      root      unknown             /proc/3/root (readlink: Permission denied)
rcu_gp                  3         root      exe       unknown             /proc/3/exe (readlink: Permission denied)
rcu_gp                  3         root      NOFD                          /proc/3/fd (opendir: Permission denied)
rcu_par_gp              4         root      cwd       unknown             /proc/4/cwd (readlink: Permission denied)
rcu_par_gp              4         root      root      unknown             /proc/4/root (readlink: Permission denied)
rcu_par_gp              4         root      exe       unknown             /proc/4/exe (readlink: Permission denied)
rcu_par_gp              4         root      NOFD                          /proc/4/fd (opendir: Permission denied)
kworker/0:0H-kblockd    6         root      cwd       unknown             /proc/6/cwd (readlink: Permission denied)
kworker/0:0H-kblockd    6         root      root      unknown             /proc/6/root (readlink: Permission denied)
kworker/0:0H-kblockd    6         root      exe       unknown             /proc/6/exe (readlink: Permission denied)
```

Run the command with root permission

```
$ sudo ./hw1 | head -n 20
COMMAND                 PID       USER      FD        TYPE      NODE      NAME
systemd                 1         root      cwd       DIR       2         /
systemd                 1         root      root      DIR       2         /
systemd                 1         root      exe       REG       1185397   /usr/lib/systemd/systemd
systemd                 1         root      mem       REG       1185397   /usr/lib/systemd/systemd
systemd                 1         root      mem       REG       1186431   /usr/lib/x86_64-linux-gnu/libm-2.31.so
systemd                 1         root      mem       REG       1186938   /usr/lib/x86_64-linux-gnu/libudev.so.1.6.17
systemd                 1         root      mem       REG       1186944   /usr/lib/x86_64-linux-gnu/libunistring.so.2.1.0
systemd                 1         root      mem       REG       1186167   /usr/lib/x86_64-linux-gnu/libgpg-error.so.0.28.0
systemd                 1         root      mem       REG       1186380   /usr/lib/x86_64-linux-gnu/libjson-c.so.4.0.0
systemd                 1         root      mem       REG       1185698   /usr/lib/x86_64-linux-gnu/libargon2.so.1
systemd                 1         root      mem       REG       1185923   /usr/lib/x86_64-linux-gnu/libdevmapper.so.1.02.1
systemd                 1         root      mem       REG       1180712   /usr/lib/x86_64-linux-gnu/libuuid.so.1.3.0
systemd                 1         root      mem       REG       1184837   /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
systemd                 1         root      mem       REG       1185810   /usr/lib/x86_64-linux-gnu/libcap-ng.so.0.0.0
systemd                 1         root      mem       REG       1185926   /usr/lib/x86_64-linux-gnu/libdl-2.31.so
systemd                 1         root      mem       REG       1186639   /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.0
systemd                 1         root      mem       REG       1186696   /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
systemd                 1         root      mem       REG       1185701   /usr/lib/x86_64-linux-gnu/liblzma.so.5.2.4
systemd                 1         root      mem       REG       1186426   /usr/lib/x86_64-linux-gnu/liblz4.so.1.9.2
```