Proc Filesystem

CSCI 3753: Operating Systems
Fall 2017

1. Introduction

The **proc filesystem** (**procfs**) is a special filesystem in Unix-like operating systems that presents information about processes and other system information in a hierarchical file-like structure, providing a more convenient and standardized method for dynamically accessing process data held in the kernel than traditional tracing methods or direct access to kernel memory. The proc filesystem provides a method of communication between kernel space and user space.

The /proc directory contains *virtual files*. These files are listed, but don't actually exist on disk; the operating system creates them on the fly if you try to read them.
Most virtual files always have a current timestamp, which indicates that they are constantly being kept up to date. The /proc directory itself is created every time you boot your system. You need to work as root to be able to examine the whole directory; some of the files (such as the process-related ones) are owned by the user who launched it. Although almost all the files are read-only, a few writable ones (notably in /proc/sys) allow you to change kernel parameters. (Of course, you must be careful if you do this.) The /proc directory is organized in virtual directories and subdirectories, and it groups files by similar topic.

2. /proc directory structure

Go to /proc in your system and list all files in that directory:

cd /proc
ls –l

The /proc directory is organized in virtual directories and subdirectories, and it groups files by similar topic. The numbered directories correspond to each running process; a special self symlink points to the current process. Some virtual files provide hardware information, such as /proc/cpuinfo, /proc/meminfo, and /proc/interrupts. Others give file-related info, such as /proc/filesystems or /proc/partitions. The files under /proc/sys are related to kernel configuration parameters.

You can view the content of these file using *cat, less* or *more* commands. Look at the contents of some of the files in this directory.

- /proc/cmdline: shows the parameters passed to the kernel at the time it is started

- /proc/cpuinfo: identifies the type of processor used by your system
- /proc/crypto: lists all installed cryptographic ciphers used by the Linux kernel
- /proc/devices: displays the various character and block devices currently configured
- /proc/filesystems: displays a list of the file system types currently supported by the kernel
- /proc/iomem: shows you the current map of the system's memory for each physical device

3. Process information

The numerical named directories represent all running processes – their directory names are process ids. When a process ends, its /proc directory disappears automatically. If you check any of these directories while they exist, you will find plenty of files, such as:

- **cmdline**: Contains the command that started the process, with all its parameters.
- **cwd**: A symlink to the current working directory (CWD) for the process; exe links to the process executable, and root links to its root directory.
- **environ**: Shows all environment variables for the process.
- **fd**: Contains all file descriptors for a process, showing which files or devices it is using.
- **maps, statm, and mem**: Deal with the memory in use by the process.
- **stat and status**: Provide information about the status of the process, but the latter is far clearer than the former.

4. Utilities

Several well-known utilities access the /proc directory to get their information, e.g. ps, uname, uptime, vmstat, top, pgrep, etc. You can create your own utilities using this filesystem. For example, the following C program prints process ids, process command and status of all processes currently running. Note that this is a subset of information displayed by utilities such as *ps –ef*.

Compile and run this program. Make sure that you understand every line of this code.

```
#include <stdio.h>
#include <dirent.h>

void print_status(long tgid) {
    char path[40], path1[40], line[100], *p;
    int cn = 0, i;
    FILE* statusf;
    FILE* statf;
```

```
    snprintf(path, 40, "/proc/%ld/status", tgid);
    snprintf(path1, 40, "/proc/%ld/stat", tgid);

    statusf = fopen(path, "r");
    if(!statusf) return;
    statf = fopen(path1, "r");
    if(!statf) return;


    printf ("%6d ", tgid);
    fgets (line, 100, statf);
    while (line[cn] != '(') cn++;
    cn++;
    for (i = 0; line[cn] != ')'; i++, cn++)
        printf ("%c", line[cn]);

    while (fgets(line, 100, statusf)) {
        if (strncmp(line, "State:", 6) != 0) continue;
        // Ignore "State:" and whitespace
        p = line + 7;
        while(isspace(*p)) ++p;
        printf(" %s", p);
        break;
    }

    fclose(statusf);
    fclose(statf);
}

int main() {
    DIR* proc = opendir("/proc");
    struct dirent* ent;
    long tgid;

    if (proc == NULL) {
        perror("opendir(/proc)");
        return 1;
    }

    while (ent = readdir(proc)) {
        if (!isdigit(*ent->d_name)) continue;
        tgid = strtol(ent->d_name, NULL, 10);
        print_status(tgid);
    }

    closedir(proc);
}
```

References

1. https://en.wikipedia.org/wiki/Procfs
2. https://linux.die.net/lkmpg/x710.html
3. https://www.linux.com/news/discover-possibilities-proc-directory