Operating Systems Lab
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320232
Date: 2016-09-01
Deadline: 2016-09-07

Problem Sheet #1

Problem 1.1: *library and system calls of* `/bin/date` (1+1+2 = 4 points)

Answer the following questions. Provide enough context information to make your results reproducible.

a) How many system calls and how many library calls does executing `/bin/date` produce?

b) What are the most frequent (top three) library and system calls and what do these system calls do?

c) What are the most time consuming library calls? Is the time consumption always the same foreach execution? Try to explain the behaviour that you observe by looking at both the library calls and the system calls in the order of their invocations.

Problem 1.2: *processing* `/bin/ps` *output* (1 point)

The Linux command `/bin/ps aux` lists all processes and kernel threads running on a Linux system. The `-T` option in addition lists user-space threads. Kernel threads do not own memory and their name is shown in square brackets in the `COMMAND` column. Write a robust shell command that filters out kernel threads from the list of processes produced by the Linux command `/bin/ps aux`.

Problem 1.3: *process listing using the* `procfs` *filesystem* (5 points)

The Linux kernel exposes information about its internal state via the `procfs` filesystem, which is typically mounted on the `/proc` directory. The various directories and files in the `procfs` filesystem tree expose kernel data structures. In particular, there is a directory for every task (process and (kernel) thread) named by a task identifier and within each directory you can find a couple of files that provide information about the task. Information about the threads of a process is available in a sub-directory.

Write a C program `mps` that produces a short process list. The process list should have the following columns:

- `PID` - the process id (or task id for kernel threads)
- `NTH` - the number of threads of the process
- `CMD` - the command line (where present)

Your program should *not* display information about kernel threads.

```
$ ./mps | head -11
    PID NTH COMMAND
      1   1 /lib/systemd/systemd --system --deserialize 17
    138   1 /lib/systemd/systemd-journald
    413   1 /usr/sbin/atd -f
    461   1 /usr/sbin/acpid
    464   1 /sbin/agetty --noclear tty1 linux
   1955   1 pickup -l -t fifo -u -c
   2031   1 smtpd -n smtp -t inet -u -c -o stress=
   2032   1 proxymap -t unix -u
   2033   1 anvil -l -t unix -u -c
  12720   4 /usr/sbin/rsyslogd -n
```

The command line option `-v` should expose details about the threads of a process:

```
$ ./mps | grep rsyslogd | grep -v grep
   12720   4 /usr/sbin/rsyslogd -n
$ ./mps -v | grep rsyslogd | grep -v grep
   12720   4 /usr/sbin/rsyslogd -n
   12720   - [/usr/sbin/rsyslogd -n]
   12723   - [/usr/sbin/rsyslogd -n]
   12724   - [/usr/sbin/rsyslogd -n]
   12725   - [/usr/sbin/rsyslogd -n]
```

While it is possible to implement this program by iterating over the various files in the `procfs` filesystem, it is much easier to use the `libprocps` library, which provides a high-level and simple to use API. It is strongly recommended to use this library for solving this problem.