Домашнее задание #5

Изучение команды **ps**

1. Выведите информацию обо всех процессах в системе, используя стандартный синтаксис:

```
$ ps -e
$ ps -ef
$ ps -eF
$ ps -ely
```

2. Выведите информацию обо всех процессах в системе, используя синтаксис BSD:

```
$ ps ax
$ ps axu
```

3. Выведите информацию обо всех процессах в системе в виде дерева:

```
$ ps -ejH
$ ps axjf
```

4. Выведите информацию о легковесных процессах:

```
$ ps -eLf
$ ps axms
```

5. Выведите атрибуты безопасности процессов:

```
$ ps -eo euser,ruser,suser,fuser,f,comm,label
$ ps axZ
$ ps -eM
```

6. Выведите информацию о процессах, выполняющихся от имени пользователя root (real & effective ID), в пользовательском формате:

```
$ ps -U root -u root u
```

7. Выведите информацию о процессах в пользовательском формате:

```
$ ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
$ ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
$ ps -Ao pid,tt,user,fname,tmout,f,wchan
```

8. Выведите только идентификаторы процессов с именем systemd:

```
$ ps -C systemd -o pid=
```

Выполнение процесса в основном и фоновом режимах

1. Откройте два окна терминала (*левый* и *правый*).

2. В левом окне запустите бесконечный процесс, который с интервалом в одну секунду будет писать строку AAA в файл **test.out** в домашнем каталоге.

```
$ ( while true; do printf "AAA %d " $$ >> ~/test.out; sleep 1; done )
```

3. В правом окне наблюдайте за работой процесса.

```
$ tail -f ~/test.out
```

4. В левом окне остановите процесс. BASH вернет номер задания в квадратных скобках. Убедитеь, что процесс перестал писать в файл.

```
$ Ctrl+z
```

5. В левом окне выведите список заданий. Знаком **+** помечено текущее задание. Определите состояние процесса. Запустите задание в фоновом режиме. Убедитеь, что процесс снова продолжил писать в файл. Определите состояние процесса ещё раз.

```
$ jobs
$ ps j
$ bg
$ jobs
$ ps j
```

6. В левом окне запустите второй бесконечный процесс в фоновом режиме, который с интервалом в одну секунду будет писать строку UUU в файл **test.out** в домашнем каталоге. Команда должна быть заключена в круглые скобки и заканчиваться знаком **&**.

```
$ ( while true; do printf "UUU %d " $$ >> ~/test.out; sleep 1; done )
```

7. В левом окне выведите список заданий. Убедитеь, что теперь два процесса пишут в файл.

```
$ jobs
```

8. В левом окне завершите процессы.

```
$ fg %номер
$ Ctrl+c
$ fg %номер
$ Ctrl+c
```

9. В правом окне завершите команду tail.

Посылка сигналов процессам

1. Откройте два окна терминала (*левый* и *правый*).

2. В левом окне запустите два бесконечных процесса, которые с интервалом в одну секунду будут писать строки в файл **test.out** в домашнем каталоге. Выведите список заданий (наблюдайте **Running**). Определите состояние процессов.

```
$ ( while true; do printf "AAA %d " $$ >> ~/test.out; sleep 1; done ) &
$ ( while true; do printf "UUU %d " $$ >> ~/test.out; sleep 1; done ) &
$ jobs
$ ps j
```

3. В правом окне наблюдайте за работой процессов.

```
$ tail -f ~/test.out
```

4. В левом окне остановите первый процесс (задание "AAA"). Выведите список заданий (наблюдайте **Stopped**).

```
$ kill -SIGSTOP %1
$ jobs
```

5. В левом окне завершите второй процесс (задание "UUU"). Выведите список заданий (наблюдайте **Terminated**).

```
$ kill -SIGTERM %2
$ jobs
```

6. В левом окне завершите процесс.

```
$ fg
$ Ctrl+c
```

7. В правом окне завершите команду tail.

Изучение возможностей файловых систем **procfs**

1. Изучите содержимое файла **/proc/version** и сравните с выводом команды **uname -a**.
2. Изучите и сравниет содержимое файлов **/proc/meminfo** и **/sys/devices/system/node/node0/meminfo**, и вывод команды **free**.
3. Изучите содержимое файла **/proc/cpuinfo**. Определите количество ядер.
4. Изучите содержимое файла **/proc/uptime** и сравните с выводом команды **uptime**.
5. Изучите специальную символьную ссылку **/proc/self**, которая указывает на подкаталог текущего процесса. В переменной **$** хранится PID текущего процесса. Объясните результаты.

```
$ echo $$
3203
$ ls -l /proc/self
lrwxrwxrwx. 1 root root 0 Oct 13 21:42 /proc/self -> 12485
$ ls -ld /proc/$$
dr-xr-xr-x. 9 vagrant vagrant 0 Oct 13 18:58 /proc/3203
```

6. Изучите содержимое файла **/proc/PID_процесса/stat**. Формат вывода можно посмотреть в исходном коде ядра Linux в файле /fs/proc/array.c в функции do_task_stat().

```
$ read pid tcomm state other < /proc/$$/stat
$ echo "Процесс $pid $tcomm находится в состоянии $state"
Процесс 3203 (bash) находится в состоянии R
```

Оформление результатов

Сохраните результат выполнения следующих команд в файл `task5.txt`:

- `history`

Загрузите данный файл на платформу для обучения.