



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт
Кафедра

компьютерных наук
автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по операционным системам Linux

«Процессы и управление ими в операционной системе Linux»

Студент

ПИ-22-1

подпись, дата

Борисов А.В.

Руководитель

подпись, дата

Кургасов В.В.

Липецк, 2024 г.

Оглавление

Цель работы.....	3
Ход работы.....	3
Часть I.....	3
Часть II.....	9
Часть III.....	15
Часть IV.....	17
Вывод.....	20
Контрольные вопросы.....	21

Цель работы

Ознакомиться на практике с понятием процесса в операционной системе. Приобрести опыт и навыки управления процессами в операционной системе Linux.

Ход работы

Часть I

1. Войти в систему под пользовательской учётной записью (не root).
(На рисунке 1)
2. Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.

Версия Linux 6.1.0-25. (Пример на рисунке 1)

3. Посмотреть процессы `ps -f`. Прокомментировать, изучив предварительно справку командой `man ps`.

Результат вызова «`ps -f`» предоставлен на рисунке 1.

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 29 12:00:53 MSK 2024 on tty1
exerted@exerte:~$ cd /
exerted@exerte:/$ ls -la vmlinuz
lrwxrwxrwx 1 root root 27 окт  1 13:46 vmlinuz -> boot/vmlinuz-6.1.0-25-amd64
exerted@exerte:/$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
exerted       550      545  1 12:02 tty1      00:00:00 -bash
exerted       555      550 99 12:03 tty1      00:00:00 ps -f
```

Рисунок 1: Вход, поиск ядра, вывод ps

В результате для каждого из процессов оболочки получаем следующую информацию:

- UID — имя пользователя вызвавшего процесс;
- PID — ID процесса;
- PPID — ID родительского процесса;
- C — Использование CPU процессором;
- STIME — Время запуска процесса;

- TTY — терминал к которому подключен процесс;
 - TIME — время работы процесса;
 - CMD — команда, породившая процесс;
4. Написать с помощью редактора vi два сценария loop и loop2.

Подтверждение создания файлов на рисунке 2.

```
exerted@exerte:~$ cat loop
while true; do true; done
exerted@exerte:~$ cat loop2
while true; do true; echo 'Hello'; done
exerted@exerte:~$
```

Рисунок 2: Созданные скрипты

5. Запустить loop2 на переднем плане: sh loop2.
6. Остановить, пошлав сигнал STOP.
7. Посмотреть последовательно несколько раз ps -f. Записать сообщение, объяснить.

После передачи сигнала остановки значение C меняется с 3 до 1. Между вызовами было 1 секунда

```
Hello
Hello
Hello
Hello
^Z
[1]+  Остановлен      sh loop2
exerted@exerte:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
exerted       550      545  0  12:02 tty1          00:00:01 -bash
exerted       560      550  3  12:07 tty1          00:00:00 sh loop2
exerted       561      550  99  12:07 tty1          00:00:00 ps -f
exerted@exerte:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
exerted       550      545  0  12:02 tty1          00:00:01 -bash
exerted       560      550  2  12:07 tty1          00:00:00 sh loop2
exerted       562      550  99  12:07 tty1          00:00:00 ps -f
exerted@exerte:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
exerted       550      545  0  12:02 tty1          00:00:01 -bash
exerted       560      550  1  12:07 tty1          00:00:00 sh loop2
exerted       563      550  99  12:08 tty1          00:00:00 ps -f
exerted@exerte:~$
```

Рисунок 3: Запуск, остановка скрипта loop2

8. Убить процесс loop2, послав сигнал kill -9 PID. Записать сообщение. Прокомментировать.

После передачи сигнала SIGKILL процесс был прекращён. (продемонстрировано на рисунке 4).

```
UID      PID     PPID    C  STIME TTY          TIME CMD
exerted   550      545    0  12:02 tty1        00:00:01 -bash
exerted   560      550    1  12:07 tty1        00:00:00 sh loop2
exerted   563      550   99  12:08 tty1        00:00:00 ps -f
exerted@exerte:~$ kill -9 560
[1]+  Убито                  sh loop2
```

Рисунок 4: Завершение скрипта loop2

9. Запустить в фоне процесс loop: sh loop&. Не останавливая, посмотреть несколько раз: ps -f. Записать значение, объяснить.

Был запущен процесс в фоновом режиме. Нагрузка C со временем начало спадать. Промежутки вызовов ps -f - 1 секунда.

```
exerted@exerte:~$ sh loop &
[1] 613
exerted@exerte:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
exerted   550      545    0  12:02 tty1        00:00:01 -bash
exerted   613      550   95  12:15 tty1        00:00:03 sh loop
exerted   614      550   78  12:15 tty1        00:00:00 ps -f
exerted@exerte:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
exerted   550      545    0  12:02 tty1        00:00:01 -bash
exerted   613      550   89  12:15 tty1        00:00:06 sh loop
exerted   615      550   77  12:15 tty1        00:00:00 ps -f
exerted@exerte:~$ ps -f
UID      PID     PPID    C  STIME TTY          TIME CMD
exerted   550      545    0  12:02 tty1        00:00:01 -bash
exerted   613      550   85  12:15 tty1        00:00:07 sh loop
exerted   616      550   74  12:15 tty1        00:00:00 ps -f
```

Рисунок 5: Запуск loop в фоне

10. Завершить процесс loop командой kill -15 PID. Записать сообщение, прокомментировать.

Процесс был остановлен сигналом SIGTERM(15).

Результат на рисунке 6.

```

UID      PID      PPID    C  STIME TTY          TIME CMD
exerted  550      545    0 12:02 tty1        00:00:01 -bash
exerted  613      550   85 12:15 tty1        00:00:07 sh loop
exerted  616      550   74 12:15 tty1        00:00:00 ps -f
exerted@exerte:~$ kill -15 613
[1]+  Завершено      sh loop

```

Рисунок 6: Завершение процесса loop

11. Третий раз запустить в фоне. Не останавливая, убить командой kill -9 PID.

В результате передачи сигнала SIGKILL процессу 623 он был прекращён.

```

exerted@exerte:~$ sh loop &
[1] 625
exerted@exerte:~$ kill -9 625
exerted@exerte:~$ ps -f
UID      PID      PPID    C  STIME TTY          TIME CMD
exerted  550      545    0 12:02 tty1        00:00:01 -bash
exerted  626      550   99 12:22 tty1        00:00:00 ps -f
[1]+  Убито          sh loop

```

Рисунок 7: Запуск и завершение скрипта loop

12. Запустить ещё один экземпляр оболочки: bash.
13. Запустить несколько процессов в фоне. Останавливать их и снова запускать. Записать результаты просмотра командой ps -f.

Были запущены несколько фоновых процессов с PID 629-632. Данные процессы и вывод после их остановки предоставлены на рисунке 8.

При остановке процесса 630 с помощью сигнала SIGSTOP его время работы перестало расти, а процесс нагрузки процессора стал уменьшаться.

На рисунке 9 изображено возобновление процесса 915 с помощью сигнала SIGCONT, в следствии чего время эффективной работы процессора снова начало расти.

```

exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      550      545   0  12:02 tty1        00:00:01 -bash
exerted      627      550   0  12:22 tty1        00:00:00 bash
exerted      629      627  26  12:24 tty1        00:00:14 sh loop
exerted      630      627  25  12:24 tty1        00:00:13 sh loop
exerted      631      627  25  12:24 tty1        00:00:13 sh loop
exerted      632      627  24  12:24 tty1        00:00:12 sh loop
exerted      633      627  29  12:25 tty1        00:00:00 ps -f
exerted@exerte:~$ kill -19 630
exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      550      545   0  12:02 tty1        00:00:01 -bash
exerted      627      550   0  12:22 tty1        00:00:00 bash
exerted      629      627  27  12:24 tty1        00:00:25 sh loop
exerted      630      627  19  12:24 tty1        00:00:17 sh loop
exerted      631      627  26  12:24 tty1        00:00:23 sh loop
exerted      632      627  26  12:24 tty1        00:00:22 sh loop
exerted      634      627  36  12:26 tty1        00:00:00 ps -f
[2]+  Остановлен sh loop

```

Рисунок 8: Запуск и остановка нескольких скриптов loop

```

exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      550      545   0  12:02 tty1        00:00:01 -bash
exerted      627      550   0  12:22 tty1        00:00:00 bash
exerted      629      627  29  12:24 tty1        00:00:56 sh loop
exerted      630      627  11  12:24 tty1        00:00:21 sh loop
exerted      631      627  29  12:24 tty1        00:00:54 sh loop
exerted      632      627  29  12:24 tty1        00:00:53 sh loop
exerted      638      627  27  12:27 tty1        00:00:00 ps -f
exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      550      545   0  12:02 tty1        00:00:01 -bash
exerted      627      550   0  12:22 tty1        00:00:00 bash
exerted      629      627  29  12:24 tty1        00:00:58 sh loop
exerted      630      627  12  12:24 tty1        00:00:24 sh loop
exerted      631      627  29  12:24 tty1        00:00:57 sh loop
exerted      632      627  29  12:24 tty1        00:00:56 sh loop
exerted      639      627  29  12:28 tty1        00:00:00 ps -f

```

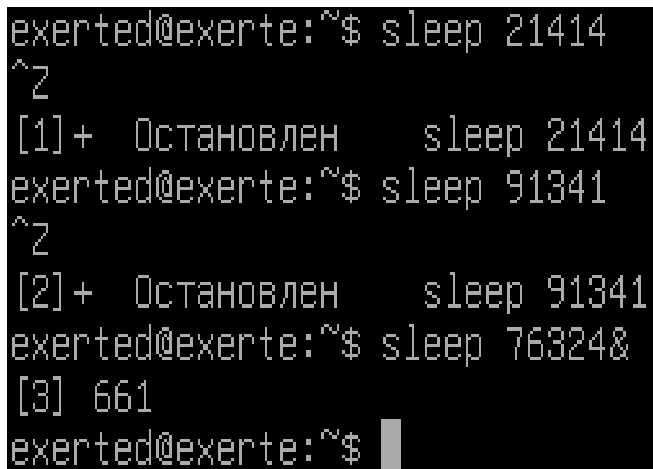
Рисунок 9: Возобновление одного из скриптов

Часть II

1. Запустить в консоли на выполнение три задачи: две в интерактивном режиме, одну - в фоновом.

При запуске задачи в интерактивном режиме невозможен ввод в терминал. Тем самым, для запуска новой интерактивной задачи первую необходимо прервать или остановить.

На рисунке 10 изображён процесс запуска двух интерактивных и одной фоновой задачи:



```
exerted@exerte:~$ sleep 21414
^Z
[1]+  Остановлен      sleep 21414
exerted@exerte:~$ sleep 91341
^Z
[2]+  Остановлен      sleep 91341
exerted@exerte:~$ sleep 76324&
[3] 661
exerted@exerte:~$
```

Рисунок 10: Задачи, запущенные в терминале

2. Перевести одну из задач, выполняющихся в интерактивном режиме, в фоновый режим.

Команда `jobs` выводит список, номера и статус всех задач, запущенных данным терминалом. Для этого используем команду `bg «номер задачи»`, была выбрана задача под номером 2. Пример вывода показан на рисунке 11.


```

exerted@exerte:~$ jobs
[1]-  Остановлен      sleep 21414
[2]+  Остановлен      sleep 91341
[3]   Запущен         sleep 76324 &
exerted@exerte:~$ bg 2
[2]+  sleep 91341 &
exerted@exerte:~$ jobs
[1]+  Остановлен      sleep 21414
[2]   Запущен         sleep 91341 &
[3]-  Запущен         sleep 76324 &
exerted@exerte:~$ █

```

Рисунок 11: Задачи, запущенные в терминале

3. Провести эксперименты по переводу задач из фонового режима в интерактивный и наоборот.

Для перехода в фоновый режим существует команда `fg`

```

exerted@exerte:~$ jobs
[1]+  Остановлен      sleep 21414
[2]   Запущен         sleep 91341 &
[3]-  Запущен         sleep 76324 &
exerted@exerte:~$ fg 2
sleep 91341
^Z
[2]+  Остановлен      sleep 91341

```

Рисунок 12: Перевод в интерактивный режим

4. Создать именованный канал для архивирования и осуществить передачу в канал:

Для создания именованного канала используется команда ``mkfifo FILENAME``. Пример создания канала предоставлен на рисунке 14.

```

exerted@exerte:~$ mkfifo pipe
exerted@exerte:~$ ls -l
итого 124
-rw-r--r-- 1 exerted exerted 10240 окт 15 02:50 1
-rw-r--r-- 1 exerted exerted 10240 окт 15 02:51 1.txt
drwxr-xr-x 5 exerted exerted 4096 окт 1 17:19 app
-rw-r--r-- 1 exerted exerted 20480 окт 15 03:03 arh1.tar
-rw-r--r-- 1 exerted exerted 20480 окт 15 02:52 arh7.tar
-rw-r--r-- 1 exerted exerted 26 окт 29 11:54 loop
-rw-r--r-- 1 exerted exerted 40 окт 29 11:55 loop2
prw-r--r-- 1 exerted exerted 0 окт 29 12:55 pipe
-rw-r--r-- 1 exerted exerted 1 окт 14 20:06 :PlugInstall
prw-r--r-- 1 exerted exerted 0 окт 29 12:53 test
-rw-r--r-- 1 exerted exerted 2781 окт 29 12:53 test_f

```

Рисунок 13: Создание именованного канала

- а) списка файлов домашнего каталога вместе с подкаталогами (ключ -R);

С помощью команды ``cat pipe > test_file &`` запустим фоновый процесс считывания содержимого канала в новый файл `test_file`.

С помощью команды ``ls -R > pipe`` передадим в именованный канал список файлов домашнего каталога с подкаталогами.

После этого считаем файл `test_file` и подтвердим, что переданные в канал данные сохранились в нем.

Результат выполнения продемонстрирован на рисунке 15. Также там заметна особенность применения именованных каналов: как только процесс, записывающий данные в канал (`ls -R > testpipe`) завершился, то и принимающий из канала данные процесс также завершился, что видно по выводам команды ``ps -f``.

```

exerted@exerte:~$ cat pipe > test_file &
[4] 676
exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      656       651  0  12:39 tty1        00:00:00 -bash
exerted      659       656  0  12:39 tty1        00:00:00 sleep 21414
exerted      660       656  0  12:40 tty1        00:00:00 sleep 91341
exerted      661       656  0  12:40 tty1        00:00:00 sleep 76324
exerted      676       656  0  12:56 tty1        00:00:00 cat pipe
exerted      677       656 72  12:56 tty1        00:00:00 ps -f
exerted@exerte:~$ ls -R > pipe
exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      656       651  0  12:39 tty1        00:00:00 -bash
exerted      659       656  0  12:39 tty1        00:00:00 sleep 21414
exerted      660       656  0  12:40 tty1        00:00:00 sleep 91341
exerted      661       656  0  12:40 tty1        00:00:00 sleep 76324
exerted      679       656 72  12:56 tty1        00:00:00 ps -f
[4]  Завершён      cat pipe > test_file

```

Рисунок 14: Использование именованного канала

б) одного каталога вместе с файлами и подкаталогами.

Аналогично предыдущему пункту создаём фоновый процесс, записывающий переданные через testpipe данные в файл test_b (cat pipe > test_file2 &) и передаём в pipe вывод команды `ls -al`. Результат проверяем с помощью команды `cat test_file2`, как показано на рисунке 16.

```

exerted@exerte:~$ cat pipe > test_file2 &
[4] 683
exerted@exerte:~$ ls -al > pipe
[4]   завершён      cat pipe > test_file2
exerted@exerte:~$ cat test_file2
итого 184
drwx----- 18 exerted exerted 4096 окт 29 13:03 .
drwxr-xr-x  9 root    root    4096 окт 15 02:07 ..
-rw-r--r--  1 exerted exerted 10240 окт 15 02:50 1
-rw-r--r--  1 exerted exerted 10240 окт 15 02:51 1.txt
drwxr-xr-x  5 exerted exerted 4096 окт  1 17:19 app
-rw-r--r--  1 exerted exerted 20480 окт 15 03:03 arh1.tar
-rw-r--r--  1 exerted exerted 20480 окт 15 02:52 arh7.tar
-rw-----  1 exerted exerted  759 окт 29 12:39 .bash_history
-rw-r--r--  1 exerted exerted  220 окт  1 14:19 .bash_logout
-rw-r--r--  1 exerted exerted 3547 окт  1 16:42 .bashrc
drwx----- 3 exerted exerted 4096 окт  1 16:50 .cache
drwxr-xr-x  3 exerted exerted 4096 окт  1 17:19 .cargo
drwx----- 3 exerted exerted 4096 окт 15 03:46 .config
-rw-----  1 exerted exerted  20 окт 29 11:39 .lessht
drwx----- 3 exerted exerted 4096 окт  1 16:50 .local
-rw-r--r--  1 exerted exerted  26 окт 29 11:54 loop
-rw-r--r--  1 exerted exerted  40 окт 29 11:55 loop2
prw-r--r--  1 exerted exerted  0 окт 29 12:56 pipe
-rw-r--r--  1 exerted exerted  1 окт 14 20:06 :PlugInstall
-rw-r--r--  1 exerted exerted  828 окт  1 16:42 .profile
drwxr-xr-x  6 exerted exerted 4096 окт  1 16:45 .rustup
-rw-r--r--  1 exerted exerted  0 окт  1 14:41 .sudo_as_admin_successful
prw-r--r--  1 exerted exerted  0 окт 29 12:53 test
-rw-r--r--  1 exerted exerted 2781 окт 29 12:53 test_f
-rw-r--r--  1 exerted exerted 2796 окт 29 12:56 test_file
-rw-r--r--  1 exerted exerted  0 окт 29 13:03 test_file2
drwxr-xr-x  2 exerted exerted 4096 окт 15 03:14 testik
-rw-r--r--  1 exerted exerted  0 окт 15 02:48 testlaba.txt
-rw-r--r--  1 exerted exerted  16 окт  1 16:51 text.txt
drwxr-xr-x  3 root    root    4096 окт 14 19:37 .vim
-rw-r--r--  1 exerted exerted  208 окт 14 20:10 .vimrc
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Видео
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Документы
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Загрузки
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Изображения
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Музыка
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Общедоступные
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Рабочий стол
drwxr-xr-x  2 exerted exerted 4096 окт 14 19:45 Шаблоны

```

Рисунок 15: Передача в файл одного каталога

Часть III

Вариант 6

1. Сгенерировать информацию — полный листинг о всех процессах системы.

а) с помощью команды ps.

Команда ps отображает информацию о процессах в момент ее вызова.

Ключи: -aux — выводит информацию о всех процессах и всех пользователей, также можно использовать ключ -eo и после можно передать данные которые нужны при отображении

Еще есть команда pstree, которая используется для иерархического представления.

```
exerted@exerte:~$ ps aux | grep sleep
exerted      490 30.7  0.2   6356   2192 tty1      S+   13:48   0:00 grep sleep
```

```
exerted@exerte:~$ ps -aux | grep nginx
exerted      488 25.0  0.2   6356   2072 tty1      S+   13:47   0:00 grep nginx
```

```
exerted@exerte:~$ pstree
systemd--agetty
        |--apache2--2*[apache2--26*[{apache2}]]
        |--cron
        |--dbus-daemon
        |--dhclient
        |--login--bash--pstree
                |      |
                |      +--3*[sleep]
                +--3*[sh]
        |--sleep
        |--sshd
        |--systemd--(sd-pam)
        |--systemd-journal
        |--systemd-logind
        |--systemd-timesyn--{systemd-timesyn}
        |--systemd-udev
```

Ps

б) С помощью команды top или htop

Плюсы использования top он является диспетчером задач и отображает информацию о процессах в реальном времени. Нтор является тем же диспетчером задач но только с интерфейсом

Main		I/O											
PID	USER	PRI	NI	VR	RES	SHR	S	CPU%	MEM%	TIME+	Command		
567	exerted	20	0	8200	4272	3172	R	4.4	0.4	0:00.26	htop		
1	root	20	0	163M	11844	9004	S	0.0	1.2	0:04.43	/sbin/init		
202	root	20	0	49392	15284	14212	S	0.0	1.6	0:01.03	/lib/systemd/systemd-journald		
226	root	20	0	26040	5748	4592	S	0.0	0.6	0:00.80	/lib/systemd/systemd-udev		
264	systemd-ti	20	0	90080	6624	5744	S	0.0	0.7	0:00.66	/lib/systemd/systemd-timesyncd		
327	systemd-ti	20	0	90080	6624	5744	S	0.0	0.7	0:00.04	/lib/systemd/systemd-timesyncd		
329	root	20	0	6608	1232	1092	S	0.0	0.1	0:00.08	/usr/sbin/cron -f		
333	messagebus	20	0	9120	4800	4228	S	0.0	0.5	0:00.43	/usr/bin/dbus-daemon --system --address=s		
340	root	20	0	16992	7684	6676	S	0.0	0.8	0:00.57	/lib/systemd/systemd-logind		
345	root	20	0	5740	3492	2664	S	0.0	0.4	0:00.09	dhclient -4 -v -i -pf /run/dhclient.ens3.		
363	root	20	0	6112	4056	3524	S	0.0	0.4	0:00.70	/bin/login -p --		
372	root	20	0	15428	8212	7016	S	0.0	0.8	0:00.22	sshd: /usr/sbin/sshd -D [listener] 0 of 1		
394	root	20	0	6564	4572	3268	S	0.0	0.5	0:00.17	/usr/sbin/apache2 -k start		
395	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.04	/usr/sbin/apache2 -k start		
396	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.04	/usr/sbin/apache2 -k start		
400	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
401	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
402	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
403	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
404	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
405	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
406	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
407	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
408	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
409	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
410	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
411	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
412	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
413	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
414	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
415	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
416	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
417	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
418	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
419	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
420	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
421	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
422	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
423	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
424	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
425	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
426	www-data	20	0	736M	11028	3292	S	0.0	1.1	0:00.00	/usr/sbin/apache2 -k start		
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit													

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
491	exerted	20	0	11644	5372	3232	R	16,7	0,5	0:00.13	top
1	root	20	0	167608	11844	9004	S	0,0	1,2	0:04.38	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	slub_flushwq
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	netns
7	root	20	0	0	0	0	I	0,0	0,0	0:00.07	kworker/0:0-cgroup_destroy
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-kblockd
9	root	20	0	0	0	0	I	0,0	0,0	0:01.74	kworker/u2:0-events_unbound
10	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_kthread
12	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_rude_kthread
13	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tasks_trace_kthread
14	root	20	0	0	0	0	S	0,0	0,0	0:00.23	ksoftirqd/0
15	root	20	0	0	0	0	I	0,0	0,0	0:01.05	rcu_preempt
16	root	rt	0	0	0	0	S	0,0	0,0	0:00.01	migration/0
17	root	20	0	0	0	0	I	0,0	0,0	0:00.64	kworker/0:1-events
18	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
20	root	20	0	0	0	0	S	0,0	0,0	0:00.02	kdevtmpfs
21	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	inet_frag_wq
22	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kauditd
23	root	20	0	0	0	0	S	0,0	0,0	0:00.00	khungtaskd
24	root	20	0	0	0	0	I	0,0	0,0	0:00.12	kworker/u2:1-flush-8:0
25	root	20	0	0	0	0	S	0,0	0,0	0:00.00	oom_reaper
26	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	writeback
27	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker/u2:2-events_unbound
28	root	20	0	0	0	0	S	0,0	0,0	0:00.10	kcompactd0
29	root	25	5	0	0	0	S	0,0	0,0	0:00.00	ksmd
30	root	39	19	0	0	0	S	0,0	0,0	0:00.00	khugepaged
31	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kintegrityd
32	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kblockd
33	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	blkcg_punt_bio
34	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	tpm_dev_wq
35	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	edac-poller
36	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	devfreq_wq
37	root	0	-20	0	0	0	I	0,0	0,0	0:00.22	kworker/0:1H-kblockd
38	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kswapd0
40	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker/u2:3-events_unbound
45	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kthrotld
47	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	acpi_thermal_pm
48	root	20	0	0	0	0	I	0,0	0,0	0:00.02	kworker/0:2-ata_sff
49	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mld

2. Завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью сигнала SIGTERM, задав его имя, второй — с помощью сигнала SIGKILL, задав его номер.

```
exerted@exerte:~$ sleep 1000 &
[1] 591
exerted@exerte:~$ tee hello &
[2] 592
exerted@exerte:~$ jobs
[1]-  Запущен          sleep 1000 &
[2]+  Остановлен      tee hello
exerted@exerte:~$ pkill -sigterm sleep
[1]-  Завершено        sleep 1000
exerted@exerte:~$ kill -s sigkill 2
-bash: kill: (2) - Операция не позволена
exerted@exerte:~$ kill -s sigkill 592
[2]+  Убито           tee hello
exerted@exerte:~$ _
```

Чтобы убить процесс по имени нужно использовать `rkill`.

Для того чтобы указать `sigkill` используем флаг `-s`.

3. Определить идентификаторы процессов, владельцем которых не является `root`

```
exerted@exerte:~$ ps -eo pid,user | awk '$2 != "root" {print $1}'
PID
264
333
395
396
461
462
588
598
599
```

Часть IV

1. Открыть окно интерпретатора команд.
 2. Вывести общую информацию о системе:
 - a) вывести информацию о текущем интерпретаторе команд;
Узнается из переменной среды: ``echo $SHELL``
 - b) вывести информацию о текущем пользователе;
Узнается из переменной среды: ``echo $USER``
 - c) вывести информацию о текущем каталоге;
Узнается из переменной среды: ``echo $PWD``
 - d) вывести информацию об оперативной памяти и области подкачки;
С помощью команды ``free -h`` (ключ `-h` переводит байты в единицы СИ)
 - e) вывести информацию о дисковой памяти.
С помощью команды ``df -h`` (ключ `-h` переводит байты в единицы СИ)
- Вывод всех перечисленных команд предоставлен на рисунке 20.


```

exerted@exerte:~$ echo $SHELL
/bin/bash
exerted@exerte:~$ echo $USER
exerted
exerted@exerte:~$ echo $PWD
/home/exerted
exerted@exerte:~$ free -h

```

	total	used	free	shared	buff/cache	available
Mem:	960Mi	230Mi	652Mi	756Ki	215Mi	730Mi
Swap:	974Mi	0B	974Mi			

```

exerted@exerte:~$ df -h
Файловая система  Размер  Использовано  Дост  Использовано%  Смонтировано в
udev              462M      0             462M      0% /dev
tmpfs             97M      416K          96M      1% /run
/dev/sda1         31G      3,1G          26G      11% /
tmpfs             481M      0            481M      0% /dev/shm
tmpfs             5,0M      0             5,0M      0% /run/lock
tmpfs             97M      0             97M      0% /run/user/1000

```

Рисунок 16: Общая информация о системе

3. Выполнить команды получения информации о процессах:

а) получить идентификатор текущего процесса(PID); - `echo \$\$`

б) получить идентификатор родительского процесса(PPID);

С помощью команды `echo \${PPID}`

с) получить идентификатор процесса инициализации системы;

Команда `ps h -eo pid | head -1`

- h: ключ, отключающий вывод заголовка;
- -e: ключ, означающий вывод всех процессов в системе;
- -o pid: ключ с аргументов, обозначающий пользовательский вывод только значений PID для каждого процессами
- | head -1 — принимает вывод команды `ps h -eo pid` и возвращает первую строку — PID первого процесса.

д) получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд - `ps -f`

е) отобразить все процессы - `ps -e`

Результат выполнения данных команд приведён на рисунке 21.

```

exerted@exerte:~$ echo $$
588
exerted@exerte:~$ echo ${PPID}
583
exerted@exerte:~$ ps h -eo pid | head -1
1
exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted       588      583   0  14:02 tty1        00:00:00 -bash
exerted       606      588  99  14:13 tty1        00:00:00 ps -f
exerted@exerte:~$ ps -e

```

4. Выполнить команды управления процессами:

- a) получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе - `ps -f`;
- b) определить текущее значение nice по умолчанию - `nice` без аргументов;
- c) запустить интерпретатор bash с понижением приоритета - `nice -n 10 bash`;
- d) определить PID запущенного интерпретатора - `echo \$\$`;
- e) установить приоритет запущенного интерпретатора равным 5 `su -c „renice -n 5 “` (`su -c` так как требуется доступ от имени привилегированного пользователя);
- f) получить информацию о процессах bash - `ps lax | grep bash`.

Пример выполнения данных команд приведён на рисунке 22. Как можно заметить, новый терминал и вызванная в нем команда `grep bash` имеют приоритет 5.

```

exerted@exerte:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
exerted      649       644  6  14:18 tty1      00:00:00 -bash
exerted      653       649 99  14:18 tty1      00:00:00 ps -f
exerted@exerte:~$ nice
0
exerted@exerte:~$ nice -n 10 bash
exerted@exerte:~$ nice
10
exerted@exerte:~$ echo $$
655
exerted@exerte:~$ su -c 'renice -n 5 655'
Пароль:
655 (process ID) old priority 10, new priority 5
exerted@exerte:~$ ps lax | grep bash
4  1000    649      644  20   0   7972  4624 do_wai S    tty1      0:00 -bash
0  1000    655      649  25   5   8004  4700 do_wai SN  tty1      0:00 bash
0  1000    661      655  25   5   6356  2176 pipe_r SN+  tty1      0:00 grep bash
exerted@exerte:~$

```

Рисунок 17: Результат выполнения команд управления процессами

Вывод

В результате выполнения данной лабораторной работы я научился взаимодействовать и управлять процессами в операционной системе Linux. Я научился посылать сигналы прерывания процессами и переводить их из интерактивного режима в фоновый и обратно в зависимости от того, какие функции они должны выполнять в системе.

Контрольные вопросы

1. Перечислите состояния задачи в ОС Linux.

- Runnable (R) — процесс выполняется/готов к выполнению;
- Sleeping (S) — процесс находится в ожидании события;
- Uninterruptible (D) — аналогичен предыдущему, однако не прерывается для обработки сигналов;
- Stopped (T) — процесс остановлен;
- Zombie (Z) — процесс завершил выполнение и почти полностью выгружен из памяти, хранит только код завершения.

2. Как создаются задачи в ОС Linux?

При помощи команды `fork()` дублируется текущий процесс, после чего в памяти размещаются данные о новом, дочернем для текущего, процессе.

При вызове новой команды в фоновом режиме процесс командной оболочки дублируется с помощью `fork`, и дочерний процесс с новым PID исполняет заданную команду.

3. Назовите классы потоков ОС Linux.

Пользовательские потоки — реализуются через специальные библиотеки потоков

Потоки на уровне ядра — реализуются через системные вызовы

4. Как используется приоритет планирования при запуске задачи?

В зависимости от приоритета задачи ей выделяется разное количество квантов времени. На выполнение идут в первую очередь задачи с более высоким приоритетом.

5. Объясните, что произойдёт, если запустить программу в фоновом режиме без подавления потока вывода.

Она продолжит отправлять сообщения в стандартный поток вывода.

6. Объясните разницу между действием сочетаний клавиш Ctrl^Z и Ctrl^C.

Ctrl+Z посылает SIGTSTP (20), приостанавливающий процесс *временно*.

Ctrl+C посылает SIGINT(2), вызывающий *завершение работы* процесса.

7. Опишите, что значит каждое поле вывода команды jobs.

В общем виде вывод команды jobs можно представить следующей конструкцией:

[<Id>] [+|-] <status> <command>

- <ID> - номер задачи в списке задач;
- [+|-] - обозначает «текущую» задачу — знак + означает, что задача выбрана будет параметром по умолчанию для команд fg, bg. Знак „-“ означает, что задача является следующим параметром по умолчанию после изменения текущего параметра;
- Status — текущий статус задачи (Running/Interrupted/...);
- command — команда, запущенная в виде отдельной задачи.

8. Назовите главное отличие утилиты top от ps.

`ps` выводит список процессов в момент вызова команды (CLI инструмент).

`top` выводит интерактивный список процессов (TUI инструмент).

9. В чем отличие результата выполнения команд top и htop?

`top` предустановлена с большинстве UNIX-систем и предоставляет базовый мониторинг процессов.

`htop` (как и atop, btop) является сторонней TUI утилитой и устанавливается отдельно, но взамен предлагает более удобный и понятный интерфейс.

10. Какую комбинацию клавиш нужно использовать для принудительного завершения задания, запущенного в интерактивном режиме?

Ctrl+C, посылает SIGINT

11. Какую комбинацию клавиш нужно использовать для приостановки задания, запущенного в интерактивном режиме?

Ctrl+Z, посылает SIGTSTP

12. Какая команда позволяет послать сигнал конкретному процессу?

``kill -SIG PID``, где SIG — сигнал, PID — id процесса.

13. Какая команда позволяет поменять поправку к приоритету уже запущенного процесса?

`renice -n <новый_приоритет> PID`

14. Какая команда позволяет запустить задание с пониженным приоритетом?

`nice -n <новая_программа> <команда>`

15. Какая команда позволяет запустить задание с защитой от прерывания при выходе из системы пользователя?

`nohup <команда> [ключи] &`

16. Какой процесс всегда присутствует в системе и является предком всех процессов?

Процесс init (PID 1).

17. Каким образом можно запустить задание в фоновом режиме?

Добавить знак амперсанд (&) в конце команды.

18. Каким образом задание, запущенное в фоновом режиме, можно перевести в интерактивный режим?

С помощью `fg <номер>`, где <номер> отвечает за номер задачи в списке задач jobs.

19. Каким образом приостановленное задание можно перевести в интерактивный режим?

С помощью `fg <номер>`, где `<номер>` отвечает за номер задачи в списке задач `jobs`. При переводе в интерактивный режим приостановленная задача возобновляется.

20. Что произойдёт с заданием, выполняющимся в фоновом режиме, если оно попытается обратиться к терминалу?

Задание приостановится, так как не сможет считать данные с потока ввода (см. рисунок 12).

21. Сколько терминалов может быть открыто в одной системе? Как перемещаться между терминалами (какие комбинации клавиш необходимо использовать)?

Шесть терминалов (`tty1-6`). Перемещаться между ними можно с помощью сочетаний клавиш `Ctrl+Alt+[F1-F6]`.

22. В чем отличие идентификаторов PID и PPID? При каких условиях возможна ситуация, когда PPID равен нулю или отсутствует?

PID — Уникальный идентификатор, выделяемый каждому процессу.

PPID — идентификатор процесса, породившего данный процесс. Если у процесса «нет родителя» или он преждевременно завершился, что в качестве родителя указывается `init`, `PPID=1`.

23. Поясните, от чего зависит максимальное значение PID.

По умолчанию `~32000`, т. к. значение лежит в целочисленном типе `int`. Может быть изменено на величину до 4 млн.

24. В каком случае, при создании нового процесса, его идентификатор (PID) будет меньше, чем у процесса, запущенного ранее?

В качестве PID выдаётся первое свободное значение идентификатора, начиная с последнего выданного PID-а. Если счетчик последнего PID переполнится, то поиск снова начнётся с 1 и может оказаться меньше PPID.