**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

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

Институт компьютерных наук

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине «Операционные системы Linux»

На тему «Процессы и управление ими в операционной системе Linux»

Студент ПИ-22-1 _____ Пахомов А.А.

подпись, дата

Руководитель

канд.техн.наук, доцент _____ Кургасов В.В.

ученая степень, ученое звание подпись, дата

Липецк, 2024 г.

**Оглавление**

Цель работы ..... 3
Ход работы ..... 4
1. Часть I ..... 4
2. Часть II ..... 10
3. Часть III ..... 13
4. Часть IV ..... 16
Вывод ..... 21
Контрольные вопросы ..... 22

**Цель работы**

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

**Ход работы**

**1. Часть I**

1.1. Войти под пользовательской учетной записью (не root). Найти файл с образом ядра. Выяснить по имени файла номер версии Linux.

Файл с образом ядра находится в директории boot (см. рис.1). Из рисунка 1 следует, что файлом с образом ядра является «vmlinuz-6.1.0.25-amd64». Из этого сделаем вывод, что текущая версия ядра – 6.1.0.25.


*Рисунок 1 – файлы в директории boot*

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

Команда ps -f используется для отображения списка запущенных процессов с дополнительной информацией о них. Ключ -f указывает на полный формат вывода.

Типичный вывод команды ps -f (см. рис. 2) содержит следующие столбцы:

1. UID — идентификатор пользователя, который запустил процесс.
2. PID — идентификатор процесса.
3. PPID — идентификатор родительского процесса
4. C — фактор использования процессора процессом.
5. STIME — время старта процесса.
6. TTY — терминал, с которого запущен процесс.
7. TIME — общее время, которое процесс использовал на процессоре.
8. CMD — команда или путь к программе, которая запустила процесс.

```
root@vbox:~# ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
root         501       1  0 23:03 tty1     00:00:00 /bin/login -p --
root         576     569  0 23:06 tty1     00:00:00 su root
root         577     576  0 23:06 tty1     00:00:00 bash
root         580     577  0 23:07 tty1     00:00:00 ps -f
```

*Рисунок 2 – вывод команды ps -f*

1.3. Написать с помощью редактора vi два сценария loop и loop2.

С помощью редактора vi были написаны следующие сценарии:

- while true; do true; done – для loop;
- while true; do true; echo “Hello”; done – для loop2.

Также с помощью команды chmod u+x сделаем файлы исполняемыми. Результат этих действий представлен на рисунке 3.

```
osalex@vbox:~$ chmod u+x loop
osalex@vbox:~$ chmod u+x loop2
osalex@vbox:~$ ls
loop  loop2
osalex@vbox:~$ cat loop
while true; do true; done
osalex@vbox:~$ cat loop2
while true; do true; echo "Hello"; done
```

*Рисунок 3 – создание двух сценариев*

1.4. Запустить loop2 на переднем плане.

Для запуска программы на переднем плане напишем команду sh loop2. Из-за этого в терминале каждый в новой строке будет выводится строка «Hello».

1.5. Остановить, послав сигнал STOP.

Для остановки работы программы loop2 перейдём в новый терминал сочетанием клавиш Win + →. С помощью команды ps -ef | grep loop  найдём процессы, связанные с loop2. Из рисунка 4 следует, что PID необходимого нам процесса равен 607.

```
osalex@vbox:~$ ps -ef | grep loop
osalex       607     567 92 11:41 tty1     00:00:12 sh loop2
osalex       609     577  0 11:42 tty2     00:00:00 grep loop
osalex@vbox:~$ kill -STOP 607
osalex@vbox:~$
```

*Рисунок 4 – остановка процесса*

Зная PID процесса, остановим его с помощью команды kill -STOP <PID>.

Перейдём в начальный терминал и увидим, что процесс был остановлен (см. рис. 5).

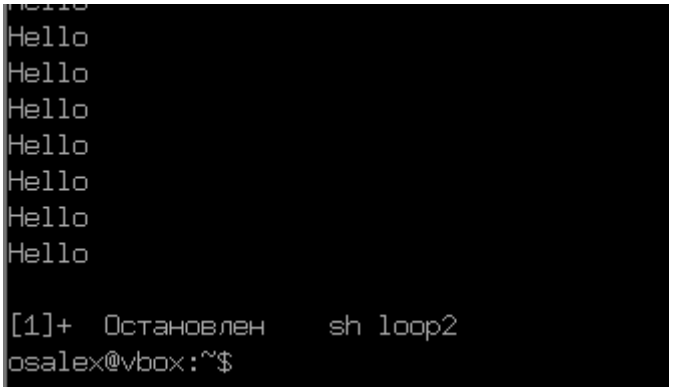



*Рисунок 5 –процесс остановлен*

1.6. Посмотреть последовательно несколько раз ps -f. Записать сообщение, объяснить.

Из рисунка 6 следует, что время, затраченное для остановленного процесса (TIME), остаётся неизменным, что свидетельствует о том, что процесс sh loop2 остановлен.

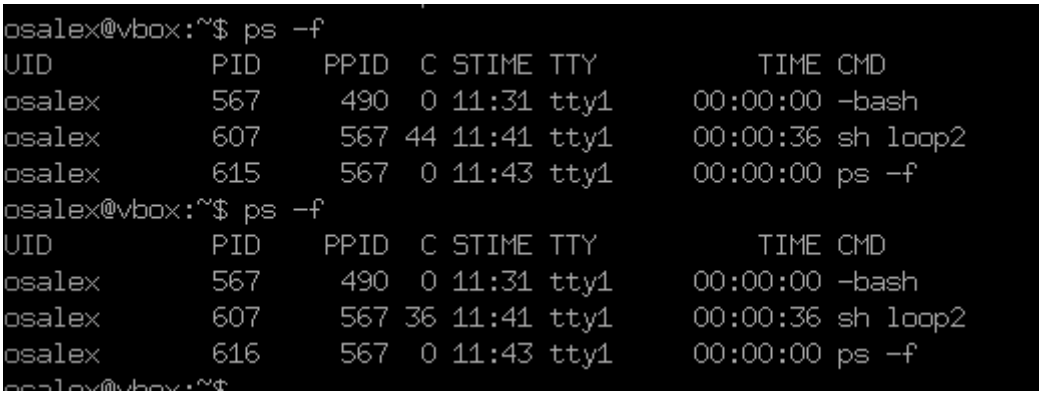



*Рисунок 6 – анализ процессов*

1.7. Убить процесс loop2, послав сигнал kill -9 PID.

Прекратим существование процесса, используя команду kill -9 607 (см. рис. 7). После чего в терминале появляется сообщение о принудительном завершении процесса.

```
osalex@vbox:~$ kill -9 607
[1]+  Убито               sh loop2
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       617     567  0 11:44 tty1     00:00:00 ps -f
```

*Рисунок 7 – принудительное завершение процесса*

-9 в команде означает код сигнала, обозначающий SIGKILL, который мгновенно завершает процесс без возможности сохранить данные или завершить операции корректно.

1.8. Запустить в фоне процесс loop: sh loop &. Не останавливая, посмотреть несколько раз ps -f.

Для запуска процесса на фоне к прошлой команде sh loop добавим знак & (см. рис. 8).

```
osalex@vbox:~$ sh loop &
[1] 618
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       618     567 99 11:45 tty1     00:00:06 sh loop
osalex       619     567  0 11:45 tty1     00:00:00 ps -f
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       618     567 99 11:45 tty1     00:00:09 sh loop
osalex       620     567  0 11:45 tty1     00:00:00 ps -f
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       618     567 99 11:45 tty1     00:01:26 sh loop
osalex       621     567  0 11:46 tty1     00:00:00 ps -f
```

*Рисунок 8 – выполнение процесса на фоне*

Из рисунка 8 следует, что интересующий нас процесс выполняется, что свидетельствует время, затраченное для процесса (TIME), которое постоянно увеличивается.

1.9. Завершить процесс loop командой kill -15 PID.

Завершим процесс командой kill -15 618 (см. рис. 9). Команда kill -15 отправляет процессу сигнал завершения SIGTERM, который завершает процесс, позволяя ему корректно освободить ресурсы и завершить свои задачи перед остановкой.

```
osalex@vbox:~$ kill -15 618
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       624     567  0 11:48 tty1     00:00:00 ps -f
[1]+  Завершено      sh loop
```

*Рисунок 9 – завершение фонового процесса*

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

Запустим опять процесс на фоне и прекратим его существование с помощью команды kill -9 (см. рис. 10). В этот раз процесс сразу принудительно завершится.

```
osalex@vbox:~$ sh loop &
[1] 631
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       631     567 99 11:49 tty1     00:00:03 sh loop
osalex       632     567  0 11:49 tty1     00:00:00 ps -f
osalex@vbox:~$ kill -9 631
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       567     490  0 11:31 tty1     00:00:00 -bash
osalex       633     567  0 11:49 tty1     00:00:00 ps -f
[1]+  Убито              sh loop
```

*Рисунок 10 – принудительное завершение фонового процесса*

1.11. Запустить еще один экземпляр оболочки: bash.

Как и в пункте 1.5 воспользуемся командой Win + → для перехода в новый терминал (см. рис. 11).

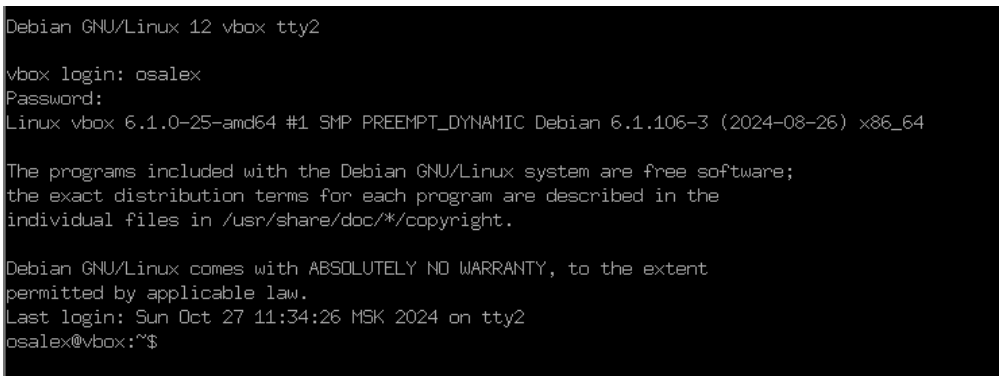



*Рисунок 11 – запуск нового терминала*

1.12. Запустить несколько процессов в фоне. Останавливать их и снова запускать. Записать результаты просмотра командой ps -f.

Запустим два фоновых процесса loop. Будем останавливать их с помощью команды kill -STOP <PID> и запускать остановленные процессы с помощью команды kill -CONT <PID>. Результат этих действий представлен на рисунке 12.

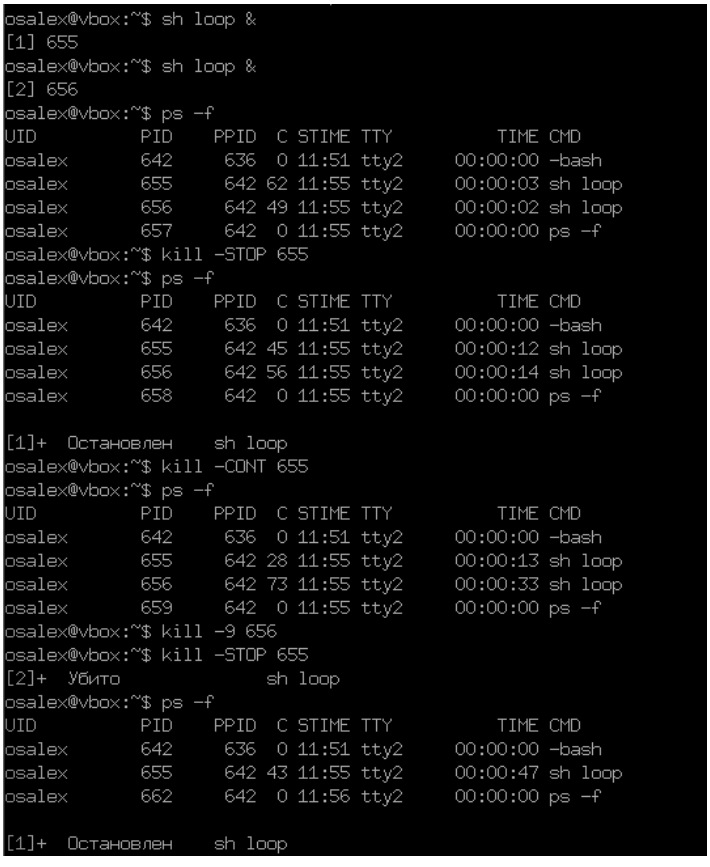



*Рисунок 12 – Исследование двух фоновых процессов*

**2. Часть II**

2.1. Запустить в консоли на выполнение три задачи: две в интерактивном режиме, одну – в фоновом.

Для данного задания создадим третий сценарий loop3, который похож на loop: while true; do true; done. Теперь запустим первый сценарий на фоне, а второй – в интерактивном режиме (см. рис. 13).


*Рисунок 13 – запуск первых двух сценариев*

Третий сценарий запустим в новом терминале в интерактивном режиме (см. рис. 14).

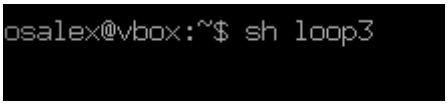



*Рисунок 14 – запуск третьего сценария*

С помощью команды, представленной на рисунке 15, рассмотрим текущие процессы.

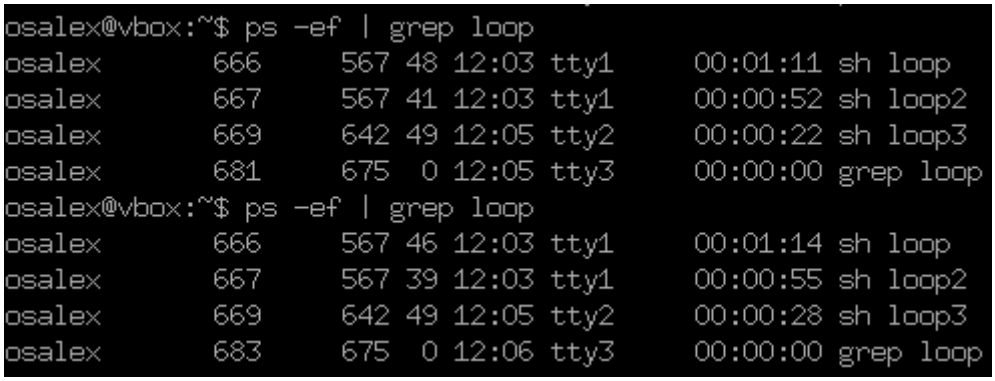



*Рисунок 15 – просмотр процессов*

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

Остановим выполнения процесса sh loop3, послав сигнад STOP (см. рис. 16).

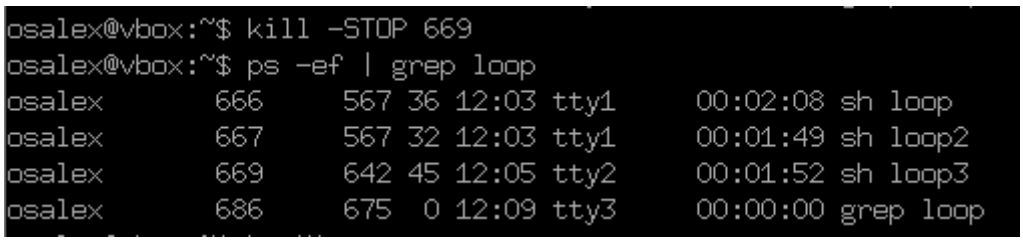



*Рисунок 16 – остановка третьего сценария*

Затем с помощью команды bg %<job_number> переводим остановленный процесс в фоновый режим (см. рис. 17).

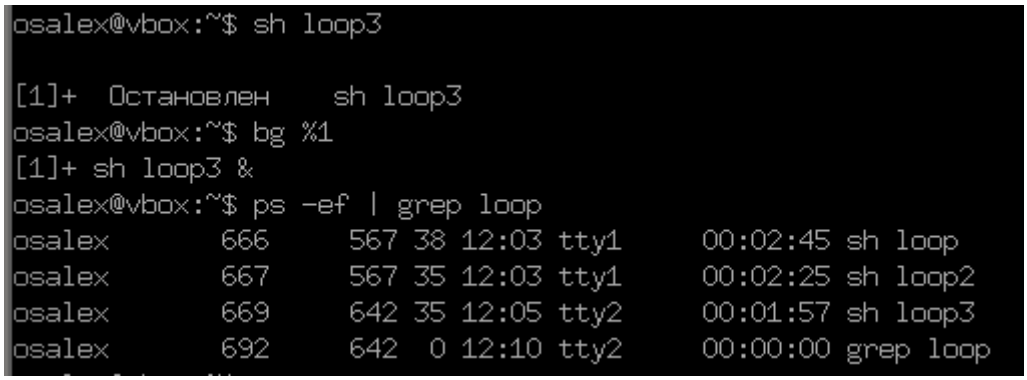



*Рисунок 17 – перевод процесса в фоновый режим*

2.3. Провести эксперименты по переводы задач из фонового режима в интерактивный и наоборот.

Для возврата процесса в фоновый режим применяется команда fg %job number. Применим данную команда для того же процесса loop3, предварительно остановив его командой kill -STOP PID (см. рис. 18).

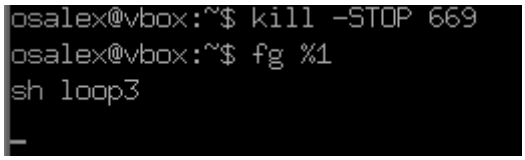



*Рисунок 18 – перевод процесса в интерактивный режим*

2.4. Создать именованный канал для архивирования и осуществить передачу в канал списка файлов домашнего каталога вместе с подкаталогами и одного каталога вместе с файлами и подкаталогами.

Создаём канал channel в домашней директории. С помощью команды ls -R ~ > channel перенаправляем поток вывода списка файлов домашней директории в к именнованный канал (см. рис. 19).

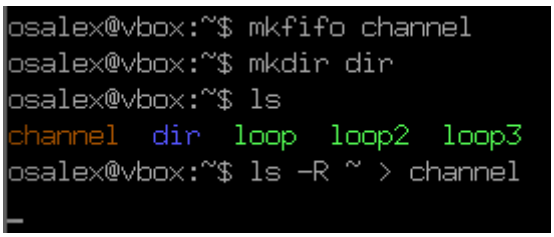



*Рисунок 19 – создание канала*

Выведем содержимое канала на рисунке 20.


*Рисунок 20 – вывод содержимого канала*

Заархивируем каталог dir из домашней директории (см. рис. 21).

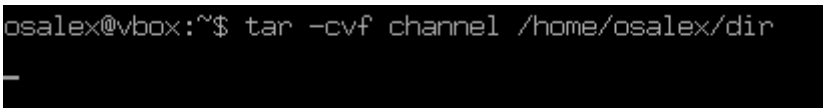



*Рисунок 21 – архивирование директории*

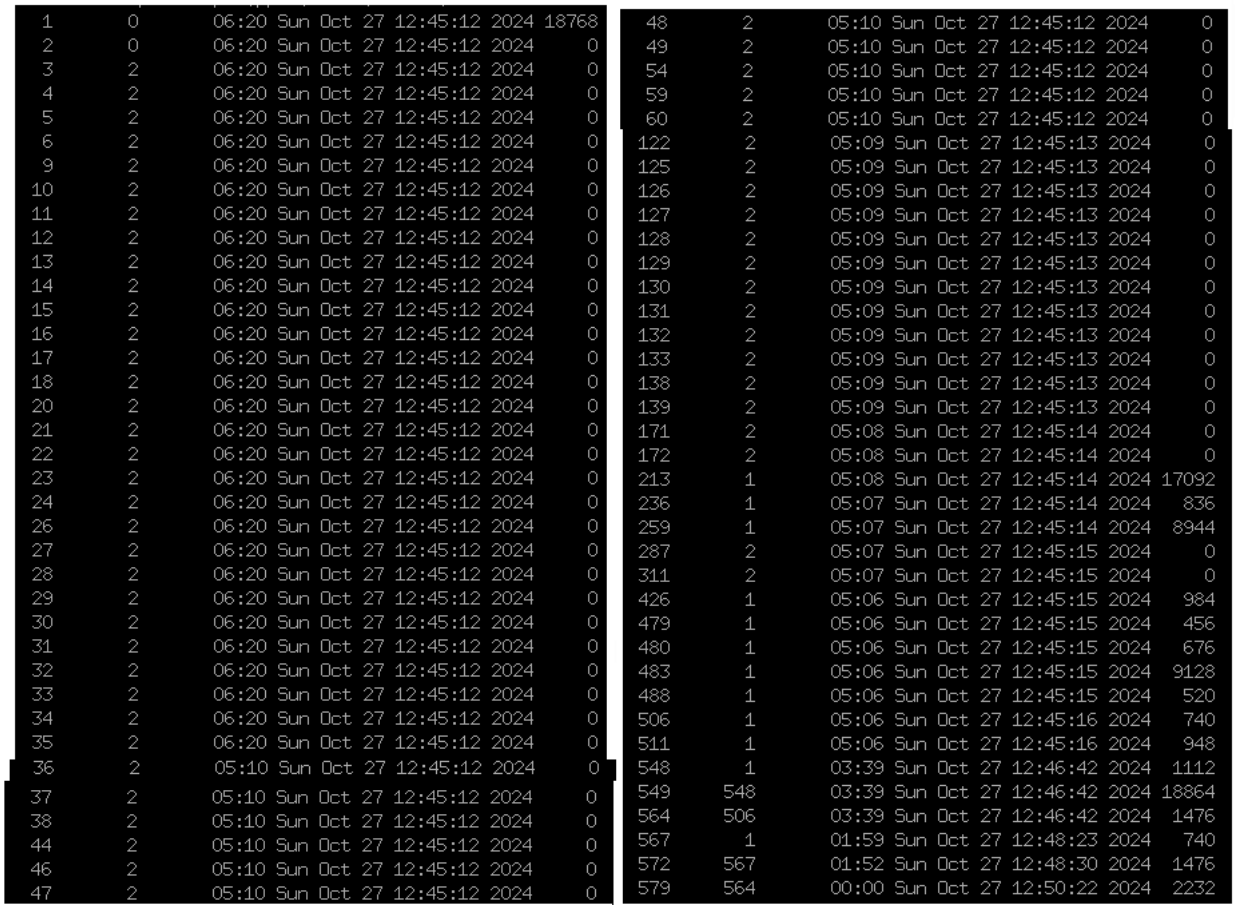
Теперь командой cat < channel > arh.tar перенаправим поток вывода канала в архив (см. рис. 22).


*Рисунок 22 – Передача данных канала в архив*

### 3. Часть III

3.1. Сгенерировать следующую информацию – полный листинг в длинном формате о процессах текущего пользователя: PID, PPID, выделенное время ЦП, время запуска, размер образа.

Для выполнения задания в терминале введём следующую команду:ps -eo pid,ppid,etime,lstart,size --user <USER> (см. рис. 23).



| | | | |
|---|---|---|---|
| 1 | 0 | 06:20 Sun Oct 27 12:45:12 2024 | 18768 |
| 2 | 0 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 3 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 4 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 5 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 6 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 9 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 10 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 11 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 12 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 13 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 14 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 15 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 16 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 17 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 18 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 20 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 21 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 22 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 23 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 24 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 26 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 27 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 28 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 29 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 30 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 31 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 32 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 33 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 34 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 35 | 2 | 06:20 Sun Oct 27 12:45:12 2024 | 0 |
| 36 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 37 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 38 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 44 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 46 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 47 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 48 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 49 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 54 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 59 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 60 | 2 | 05:10 Sun Oct 27 12:45:12 2024 | 0 |
| 122 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 125 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 126 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 127 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 128 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 129 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 130 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 131 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 132 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 133 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 138 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 139 | 2 | 05:09 Sun Oct 27 12:45:13 2024 | 0 |
| 171 | 2 | 05:08 Sun Oct 27 12:45:14 2024 | 0 |
| 172 | 2 | 05:08 Sun Oct 27 12:45:14 2024 | 0 |
| 213 | 1 | 05:08 Sun Oct 27 12:45:14 2024 | 17092 |
| 236 | 1 | 05:07 Sun Oct 27 12:45:14 2024 | 836 |
| 259 | 1 | 05:07 Sun Oct 27 12:45:14 2024 | 8944 |
| 287 | 2 | 05:07 Sun Oct 27 12:45:15 2024 | 0 |
| 311 | 2 | 05:07 Sun Oct 27 12:45:15 2024 | 0 |
| 426 | 1 | 05:06 Sun Oct 27 12:45:15 2024 | 984 |
| 479 | 1 | 05:06 Sun Oct 27 12:45:15 2024 | 456 |
| 480 | 1 | 05:06 Sun Oct 27 12:45:15 2024 | 676 |
| 483 | 1 | 05:06 Sun Oct 27 12:45:15 2024 | 9128 |
| 488 | 1 | 05:06 Sun Oct 27 12:45:15 2024 | 520 |
| 506 | 1 | 05:06 Sun Oct 27 12:45:16 2024 | 740 |
| 511 | 1 | 05:06 Sun Oct 27 12:45:16 2024 | 948 |
| 548 | 1 | 03:39 Sun Oct 27 12:46:42 2024 | 1112 |
| 549 | 548 | 03:39 Sun Oct 27 12:46:42 2024 | 18864 |
| 564 | 506 | 03:39 Sun Oct 27 12:46:42 2024 | 1476 |
| 567 | 1 | 01:59 Sun Oct 27 12:48:23 2024 | 740 |
| 572 | 567 | 01:52 Sun Oct 27 12:48:30 2024 | 1476 |
| 579 | 564 | 00:00 Sun Oct 27 12:50:22 2024 | 2232 |

*Рисунок 23 – листинг всех процессов*

Объяснение опций:

- -e: отображает все процессы.
- -o: позволяет настроить вывод с выбором нужных полей.
  - pid: идентификатор процесса.
  - ppid: идентификатор родительского процесса.
  - etime: время, в течение которого процесс был активен.

- lstart: время запуска процесса.
- size: размер образа процесса в килобайтах.

- --user <USER>: фильтрует процессы, принадлежащие текущему пользователю (значение переменной $USER).

3.2. С помощью сигнала SIGTSTP приостановить выполнение процесса, владельцем которого является текущий пользователь. Через несколько секунд возобновить выполнение процесса.

Из предыдущего пункта выбирает PID процесса и посылаем сначала сигнал SIGTSTP, останавливая процесс, а затем возобновляем процесс, посылая сигнал CONT (см. рис. 24).

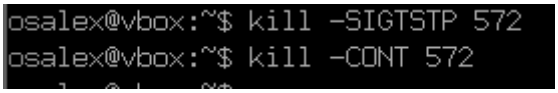



*Рисунок 24 – работа с процессом*

3.3. Определить идентификатор и имя процесса, созданного последним пользователем root.

Для выполнения задания воспользуемся комбинацией команд, представленной на рисунке 25.

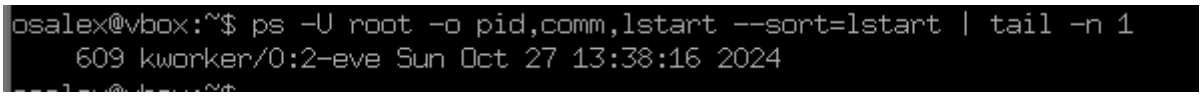



*Рисунок 25 – последний процесс*

Данная команда состоит из следующих частей:

- ps -U root: показывает процессы, запущенные пользователем root. Здесь используется флаг -U, чтобы выбрать процессы по имени пользователя.
- -o pid,comm,lstart: указывает вывод следующих полей:
- pid: идентификатор процесса.
- comm: команда (имя процесса).
- lstart: время запуска процесса.

- --sort=start_time: сортирует процессы по времени запуска (чем позже был запущен процесс, тем выше в списке).
- tail -n 1: выводит последнюю строку, что соответствует последнему процессу (самому новому).

**4. Часть IV**

4.1. Вывести общую информацию о системе

4.1.1. Вывести информацию о текущем интерпретаторе команд

Переменная SHELL хранит путь к текущему интерпретатору команд, поэтому используем команду echo (см. рис. 26).

```
osalex@vbox:~$ echo $SHELL
/bin/bash
```

*Рисунок 26 – переменная SHELL*

4.1.2. Вывести информацию о текущем пользователе

Для вывода текущего пользователя используем команду whoami (см. рис. 27).

```
osalex@vbox:~$ whoami
osalex
```

*Рисунок 27 – текущий пользователь*

4.1.3. Вывести информацию о текущем каталоге

Для вывода текущей директории используем команду pwd (см. рис. 28).

```
osalex@vbox:~$ pwd
/home/osalex
```

*Рисунок 28 – текущая директория*

4.1.4. Вывести информацию об оперативной памяти и области подкачки

Команда free -h (см. рис. 29) используется для отображения информации об использовании оперативной памяти и пространства подкачки (swap) в системе. Опция -h выводит данные в человеко-читаемом формате (например, в мегабайтах или гигабайтах), вместо байтов.

```
osalex@vbox:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           1,9Gi       200Mi       1,8Gi       552Ki        90Mi       1,7Gi
Swap:          974Mi          0B       974Mi
```

*Рисунок 29 – информация об оперативной памяти и области подкачки*

4.1.5. Вывести информацию о дисковой памяти

Команда df -h (см. рис. 30) используется для отображения информации о дисковом пространстве, доступном в файловых системах, в удобочитаемом формате (опция -h выводит данные в мегабайтах, гигабайтах и т.д.).

```
osalex@vbox:~$ df -h
Файловая система Размер Использовано  Дост Использовано% Смонтировано в
udev                965M            0  965M            0% /dev
tmpfs               197M          552K  197M            1% /run
/dev/sda1            19G          1,9G   16G           11% /
tmpfs               984M            0  984M            0% /dev/shm
tmpfs               5,0M            0  5,0M            0% /run/lock
tmpfs               197M            0  197M            0% /run/user/1000
```

*Рисунок 30 – информация о дисковой памяти*

4.2. Выполнить команды получения информации о процессах

4.2.1. Получить идентификатор текущего процесса

Для получения идентификатора текущего процесса воспользуемся командой echo $$ (см. рис. 31).

```
osalex@vbox:~$ echo $$
564
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       564     506  0 12:46 tty1     00:00:00 -bash
osalex       617     564  0 13:46 tty1     00:00:00 ps -f
```

*Рисунок 31 – идентификатор текущего процесса*

4.2.2. Получить идентификатор родительского процесса

Идентификатор родительского процесса хранится в переменной PPID. С помощью команды echo выведем эту переменную (см. рис. 32).

```
osalex@vbox:~$ echo $PPID
506
osalex@vbox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       564     506  0 12:46 tty1     00:00:00 -bash
osalex       618     564  0 13:47 tty1     00:00:00 ps -f
```

*Рисунок 32 – идентификатор родительского процесса*

4.2.3. Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Воспользуемся командой ps -fu $(whoami) (см. рис. 33). Опция -u указывает, что нужно отфильтровать процессы по пользователю. Команда $(whoami) подставляет имя текущего пользователя

```
osalex@vbox:~$ ps -fu $(whoami)
UID          PID    PPID  C STIME TTY          TIME CMD
osalex       548       1  0 12:46 ?        00:00:00 /lib/systemd/systemd --user
osalex       549     548  0 12:46 ?        00:00:00 (sd-pam)
osalex       564     506  0 12:46 tty1     00:00:00 -bash
osalex       572     567  0 12:48 tty2     00:00:00 -bash
osalex       624     564  0 13:48 tty1     00:00:00 ps -fu osalex
```

*Рисунок 33 – информация о выполняющихся процессах текущего пользователя*

4.2.4. Отобразить все процессы

Отображение всех процессов происходит благодаря команде ps -ef (см. рис. 34).

```
root           36       2  0 12:45 ?        00:00:00 [devfreq_wq]
root           37       2  0 12:45 ?        00:00:00 [kworker/0:1H-kblockd]
root           38       2  0 12:45 ?        00:00:00 [kswapd0]
root           44       2  0 12:45 ?        00:00:00 [kthrotld]
root           46       2  0 12:45 ?        00:00:00 [acpi_thermal_pm]
root           48       2  0 12:45 ?        00:00:00 [mld]
root           49       2  0 12:45 ?        00:00:00 [ipv6_addrconf]
root           54       2  0 12:45 ?        00:00:00 [kstrp]
root           59       2  0 12:45 ?        00:00:00 [zswap-shrink]
root           60       2  0 12:45 ?        00:00:00 [kworker/u3:0]
root          125       2  0 12:45 ?        00:00:00 [ata_sff]
root          126       2  0 12:45 ?        00:00:00 [scsi_eh_0]
root          127       2  0 12:45 ?        00:00:00 [scsi_tmf_0]
root          128       2  0 12:45 ?        00:00:00 [scsi_eh_1]
root          129       2  0 12:45 ?        00:00:00 [scsi_tmf_1]
root          131       2  0 12:45 ?        00:00:00 [scsi_eh_2]
root          132       2  0 12:45 ?        00:00:00 [scsi_tmf_2]
root          139       2  0 12:45 ?        00:00:00 [kworker/0:2H-kblockd]
root          171       2  0 12:45 ?        00:00:00 [jbd2/sda1-8]
root          172       2  0 12:45 ?        00:00:00 [ext4-rsv-conver]
root          213       1  0 12:45 ?        00:00:00 /lib/systemd/systemd-journald
root          236       1  0 12:45 ?        00:00:00 /lib/systemd/systemd-udevd
systemd+      259       1  0 12:45 ?        00:00:00 /lib/systemd/systemd-timesyncd
root          287       2  0 12:45 ?        00:00:00 [cryptd]
root          311       2  0 12:45 ?        00:00:00 [irq/18-vmwgfx]
root          426       1  0 12:45 ?        00:00:00 dhclient -4 -v -i -pf /run/dhclient.enp0s3.pid
root          479       1  0 12:45 ?        00:00:00 /usr/sbin/cron -f
message+      480       1  0 12:45 ?        00:00:00 /usr/bin/dbus-daemon --system --address=system
root          483       1  0 12:45 ?        00:00:00 /lib/systemd/systemd-logind
root          488       1  0 12:45 ?        00:00:00 /sbin/wpa_supplicant -u -s -O DIR=/run/wpa_sup
root          506       1  0 12:45 tty1     00:00:00 /bin/login -p --
root          511       1  0 12:45 ?        00:00:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100
osalex        548       1  0 12:46 ?        00:00:00 /lib/systemd/systemd --user
osalex        549     548  0 12:46 ?        00:00:00 (sd-pam)
osalex        564     506  0 12:46 tty1     00:00:00 -bash
root          567       1  0 12:48 tty2     00:00:00 /bin/login -p --
osalex        572     567  0 12:48 tty2     00:00:00 -bash
root          584       2  0 12:57 ?        00:00:00 [kworker/u2:1-events_unbound]
root          602       2  0 13:33 ?        00:00:00 [kworker/u2:0-flush-8:0]
root          609       2  0 13:38 ?        00:00:01 [kworker/0:2-events]
root          625       2  0 13:48 ?        00:00:00 [kworker/0:0-ata_sff]
root          626       2  0 13:53 ?        00:00:00 [kworker/0:1-ata_sff]
osalex        627     564  0 13:56 tty1     00:00:00 ps -ef
osalex@vbox:~$
```

*Рисунок 34 – отображение всех процессов*

4.3. Выполнить команды управления процессами

4.3.1. Определить текущее значение nice по умолчанию

Чтобы узнать значение nice по умолчанию достаточно запустить команду nice без параметров (см. рис. 35).


*Рисунок 35 – команда nice*

4.3.2. Запустить интерпретатор bash с понижением приоритета nice -n 10 bash

Запустим интерпретатор bash с понижением приоритета nice -n 10 bash (см. рис. 36).


*Рисунок 36 – понижение приоритета*

4.3.3. Определить PID запущенного интерпретатора

Как в пункте 4.2.1, отобразим PID запущенного интерпретатора (см. рис. 37).

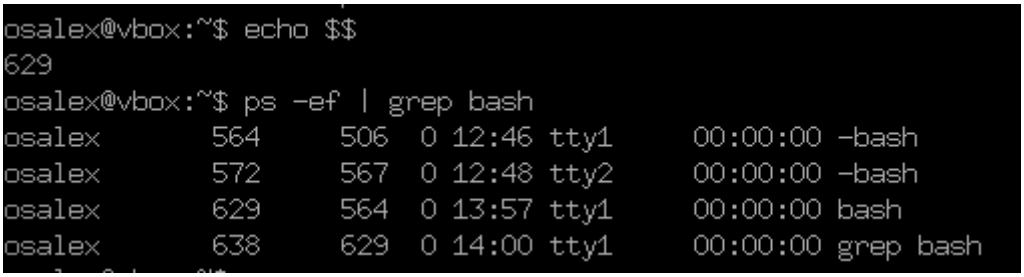



*Рисунок 37 – PID запущенного интерпретатора*

4.3.4 Установить приоритет запущенного интерпретатора равным 5

Установим приоритет запущенного интерпретатора равным 5 с помощью команды renice -n 5 629 (см. рис. 38)

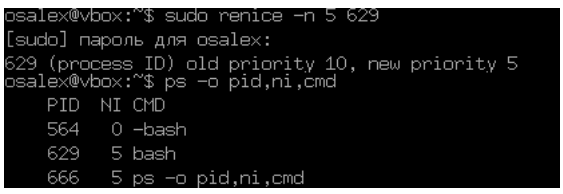



*Рисунок 38 – установка приоритета*

4.3.5. Получить информацию о процессах bash: ps lax | grep bash

Команда ps lax | grep bash используется для поиска процессов, связанных с оболочкой bash, среди всех запущенных процессов на системе (см. рис. 39).

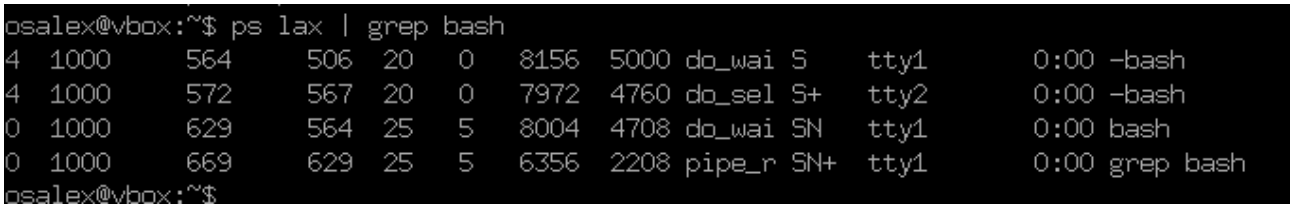



*Рисунок 39 – информация о процессах bash*

Рассмотрим команду по частям:

- ps — команда для отображения информации о запущенных процессах.
- lax — опции команды ps:
  - l — вывод в длинном формате, показывающем более подробную информацию о каждом процессе.
  - a — отображает процессы всех пользователей, не только тех, которые запущены в текущем терминале.
  - x — показывает процессы, не привязанные к терминалам, то есть фоновые процессы.
- grep bash — фильтрует вывод команды ps, отображая только те строки, которые содержат слово bash.

**Вывод**

В ходе выполнения данной лабораторной работы ознакомился на практике с понятием процесса в операционной системе. Приобрел опыт и навыки управления процессами в операционной системе Linux.

**Контрольные вопросы**

**1) Перечислите состояния задачи в ОС Linux.**

В операционной системе Linux задача (или процесс) может находится в одном из следующих состояний:

- Running— выполняется (или готов к выполнению).
- Sleeping— спит (ожидает события).
- Uninterruptible Sleep — непрерываемый сон. Процесс спит и не может быть прерван сигналом.
- Zombie — зомби-процесс. Процесс завершился, но его родительский процесс не забрал статус завершения.
- Stopped — остановлен (или на паузе).
- Dead — завершен.

**2) Как создаются задачи в ОС Linux?**

Для создания процесса в операционной системе Linux используется системный вызов fork().Этот вызов создаёт новый процесс из существующего. Существующий процесс называется родительским процессом, а созданный заново процесс — дочерним процессом. fork() возвращает родителю PID ребенка, ребенку возвращается 0.

**3) Назовите классы потоков ОС Linux.**

В ОС Linux выделяются три класса потоков для процессов:

- Потоки реального времени, обслуживаемые по алгоритмы FIFO (имеют наивысшие приоритеты и не могут вытесняться другими потоками, за исключением того же потока реального времени с более высоким приоритетом, перешедшего в состояние готовности);
- Потоки реального времени, обслуживаемые в порядке циклической очереди (имеют квант времени и могут вытесняться по таймеру. Находящийся в состоянии готовности поток выполняется в течение кванта времени, после чего поток помещается в конец своей очереди);

- Потоки разделения времени (представлены приоритетами от 100 до 139, то есть в системе Linux реализовано 140 приоритетов).

**4) Как используется приоритет планирования при запуске задачи?**

При запуске задачи приоритет планирования определяет, когда и как долго задача будет выполняться на процессоре. Зависимость приоритета от класса планирования:

- Для классов SCHED_FIFO и SCHED_RR (реального времени) — используются приоритеты от 1 до 99. Задачи с более высоким числом приоритета запускаются раньше и выполняются до завершения или блокировки (для SCHED_FIFO) либо в пределах выделенного кванта времени (для SCHED_RR).
- Для класса SCHED_OTHER (SCHED_NORMAL) — приоритет определяется значением *nice*, изменяемым от -20 до 19. Чем ниже значение nice, тем выше приоритет (выше вероятность быстрого выделения времени процессора).

**5) Объясните, что произойдет, если запустить программу в фоновом режиме без подавления потока вывода.**

Если запустить программу в фоновом режиме, но не подавить поток вывода (например, & без перенаправления >/dev/null), вывод программы все равно будет поступать в консоль. Это может привести к засорению терминала и неудобному взаимодействию.

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

- Ctrl+Z — Приостановка задачи. Эта комбинация временно останавливает текущий процесс, переводя его в состояние «приостановлен» (stopped). Процесс продолжит выполнение только после команды fg (foreground) или bg (background).
- Ctrl+C — Прерывание задачи. Эта комбинация отправляет процессу сигнал SIGINT, прерывая его выполнение. Задача завершится сразу, если она не обрабатывает или не игнорирует этот сигнал.

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

Команда jobs выводит информацию о задачах, запущенных в текущей сессии. Пример полей вывода:

- [N] — номер задачи, где N — номер задания в текущей сессии.
- Статус — показывает состояние задачи, например:
  - Running — задача выполняется.
  - Stopped — задача приостановлена.
  - Done — задача завершена.
- Команда — командная строка, которая запустила задачу.

**8) Назовите главное отличие утилиты top от jobs.**

Главное отличие заключается в области охвата и уровне детализации:

- top — отображает все процессы системы, их ресурсоемкость и состояние в реальном времени.
- jobs — отображает только задачи, запущенные в текущем сеансе оболочки, и ограничен текущим терминалом.

**9) В чем отличие результата выполнения команд top и htop?**

top и htop обе отображают процессы системы, но имеют разные интерфейсы и функции:

- top — более минималистична и предустановлена в большинстве дистрибутивов Linux. Для большинства операций требует команд через клавиатуру, предлагает базовые возможности управления процессами.
- htop — улучшенный интерфейс с возможностью использования мыши, более удобным цветовым оформлением, возможностью выбора и завершения нескольких процессов одновременно и расширенной информацией по каждому процессу (например, отображение количества потоков, температуры и использования ресурсов).

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

Ctrl+C — эта комбинация отправляет сигнал SIGINT процессу, принудительно завершая его. Она используется для немедленного завершения выполнения активного процесса.

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

Ctrl+Z — эта комбинация отправляет сигнал SIGTSTP, временно приостанавливая выполнение процесса. Процесс переводится в состояние "остановлен" (suspended) и может быть возобновлен с помощью команд fg (для работы на переднем плане) или bg (для фоновой работы).

**12) Какая команда позволяет послать сигнал конкретному процессу?**

kill — эта команда позволяет отправить сигнал конкретному процессу по его PID. Формат команды: kill -SIGNAL PID. Например, kill -9 PID отправляет сигнал SIGKILL, принудительно завершая процесс.

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

renice — команда позволяет изменить значение nice для уже работающего процесса, тем самым корректируя его приоритет. Формат: renice PRIORITY -p PID, где PRIORITY — новое значение nice (от -20 до 19).

**14) Какая команда позволяет запустить задание с пониженным приоритетом?**

nice — команда запускает новое задание с указанным значением *nice*, понижая его приоритет. Формат: nice -n PRIORITY команда, где PRIORITY — значение от -20 (высокий приоритет) до 19 (низкий приоритет).

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

nohup — команда запускает процесс, защищенный от прерывания сигналом SIGHUP, который посылается при выходе пользователя из системы. Формат: nohup команда &. Вывод команды по умолчанию будет записан в файл nohup.out.

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

В Linux процесс init всегда присутствует в системе и является предком всех процессов.

- PID 1: init всегда имеет PID (идентификатор процесса) 1.
- Это первый процесс, запускаемый ядром при загрузке системы, и он порождает все остальные процессы напрямую или через другие процессы.
- init управляет жизненным циклом всех процессов, контролируя запуск, остановку и мониторинг системных сервисов и пользовательских процессов

**17) Каким образом можно запустить задание в фоновом режиме?**

Задание можно запустить в фоновом режиме, добавив символ & в конце команды.

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

Для перевода задания из фонового режима в интерактивный используется команда:

- fg %N, где %N — номер задания. Команда fg вернет задачу в передний план и сделает ее интерактивной.

Чтобы узнать номер задания, можно воспользоваться командой jobs, которая выведет все фоновые и приостановленные задания в текущем терминале.

**19) Каким образом приостановленное задание можно перевести в интерактивный режим?**

Приостановленное задание можно вернуть в интерактивный режим с помощью команды fg %N, где %N — номер задания. Это команда возобновит выполнение приостановленного задания и переведет его в передний план.

**20) Что произойдет с заданием, выполняющимся в фоновом режиме, если оно попытается обратиться к терминалу?**

Если задание в фоновом режиме попытается обратиться к терминалу (например, запросит ввод или выведет информацию), то выполнение задачи будет приостановлено, и система отправит задаче сигнал SIGTTOU (при выводе) или SIGTTIN (при вводе). В терминале будет показано сообщение о

приостановке процесса, и его можно будет возобновить в переднем плане командой fg %N.

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

Количество терминалов: по умолчанию в большинстве Linux-систем одновременно доступны 6 виртуальных терминалов (tty1–tty6) для текстового режима. Количество доступных терминалов можно изменить в конфигурации системы.

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

PID (Process ID) — это уникальный идентификатор процесса в системе.

PPID (Parent Process ID) — это идентификатор родительского процесса.

**23) Поясните, от чего зависит максимальное значение PID?**

Максимальное значение PID зависит от конфигурации системы и определяется переменной /proc/sys/kernel/pid_max.

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

Такое возможно при перезапуске счетчика PID. Когда текущий PID достигает значения pid_max, следующий процесс начинает получать PID с минимально доступного значения (например, 1), который больше не занят.