



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: «ДПО Интаро Софт»  
на тему: «Процессы в операционной системе Linux».

Студент

AC-23-1

группа

подпись, дата

Кирнос Б. В.

фамилия, инициалы

Руководитель

к.т.н., доцент кафедры АСУ

ученая степень, ученое звание

подпись, дата

Кургасов В.В.

фамилия, инициалы

Липецк 2024

## Оглавление

Цель работы.....	2
Задачи из первой части .....	3
Задачи из второй части .....	5
Вывод .....	6
Контрольные вопросы.....	7

**Цель работы:** ознакомиться на практике с понятием процесса в операционной системе. Приобрести опыт и навыки управления процессами в операционной системе Linux.

## Вариант 5

## Задачи из первой части

1. Отобразить информацию о процессах указанного пользователя в виде иерархии, вывод отсортировать по значениям PID.

```
ps -u bazaraj -o pid,ppid,user,cmd --forest --sort=pid
```

```
bazaraj@bazara-k53sm:~$ ps -u bazaraj -o pid,ppid,user,cmd --forest --sort=pid
PID  PPID USER      CMD
2222      1 bazaraj /usr/bin/fcitx5 -d
2087    2068 bazaraj /usr/bin/startplasma-x11
2187    2087 bazaraj \_ /usr/bin/ssh-agent /usr/bin/im-launch /usr/bin/startplasma-x11
2072      1 bazaraj /usr/lib/systemd/systemd --user
2073    2072 bazaraj \_ (sd-pam)
2084    2072 bazaraj \_ /usr/bin/pipewire
2085    2072 bazaraj \_ /usr/bin/pipewire -c filter-chain.conf
2086    2072 bazaraj \_ /usr/bin/wireplumber
bazaraj@bazara-k53sm:~$
```

Рисунок 1 – Задание 1

Эта команда выводит древовидную структуру процессов пользователя bazaraj. PID - уникальный идентификатор процесса, PPID - идентификатор родительского процесса, CMD - команда/программа, которая выполняется, формат --forest - показывает иерархию процессов в виде дерева.

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

Для выполнения задания я использовал мессенджер Telegram.

```
pidof Telegram
kill -SIGSTOP 3419
kill -SIGCONT 3419
```

```
root@bazara-k53sm:/boot# pidof Telegram
3419
root@bazara-k53sm:/boot# kill -SIGSTOP 3419
root@bazara-k53sm:/boot# kill -SIGCONT 3419
root@bazara-k53sm:/boot#
```

Рисунок 2 – Задание 2

Мы сначала узнали PID телеграма, затем приостановили и запустили его обратно. Будучи остановленным, приложение не работало.

3. Определить идентификаторы и имена процессов, не связанных с указанным терминалом.

tty

```
ps -e -o pid,tty,comm | awk 'NR > 1 && $2!="pts/2" {print $1, $3}'
```

```
root@bazara-k53sm:/boot# tty
/dev/pts/2
root@bazara-k53sm:/boot# ps -e -o pid,tty,comm | awk 'NR>1 && $2!="pts/2" {print $1, $3}' >/home/pidsWithoutTerminal.txt
root@bazara-k53sm:/boot# ps -e -o pid,tty,comm | awk 'NR>1 && $2!="pts/2" {print $1, $3}'.
1  systemd
2  kthreadd
3  pool_workqueue_release
4  kworker/R-rcu_gp
5  kworker/R-sync_wq
6  kworker/R-kvfree_rcu_reclaim
7  kworker/R-slab_flushwq
8  kworker/R-netsns
9  kworker/0:0H-events_highpri
10 kworker/0:mm_percpu_wd
11 rCU_tasks_kthread
12 rCU_tasks_rude_kthread
13 rCU_tasks_trace_kthread
14 rCU_tasks_trace_kthread
15 ksoftirqd/0
16 rCU preempt
17 rCU_exp_par_gp_kthread_worker/0
18 rCU_exp_gp_kthread_worker
19 migration/0
20 idle_inject/0
21 cpuhp/0
22 cpuhp/1
23 cpuhp/2
24 cpuhp/3
25 cpuhp/4
26 migration/1
27 ksoftirqd/1
28 cpuhp/5
29 cpuhp/6
30 cpuhp/7
31 cpuhp/8
32 migration/2
33 ksoftirqd/2
34 cpuhp/9
35 kworker/2:0H-events_highpri
36 cpuhp/3
37 cpuhp/4
38 migration/3
39 ksoftirqd/3
40 cpuhp/5
41 kworker/3:0H-events_highpri
42 kdevtmpfs
```

Рисунок 3 – Задание 3

Сначала мы узнали tty активного терминала, чтобы исключить его из вывода.  
ps -e -o pid,tty,comm - показывает ВСЕ процессы системы с тремя полями:

- pid - идентификатор процесса
- tty - терминал, к которому привязан процесс (если есть)
- comm - имя команды (исполняемый файл)

awk 'NR > 1 && \$2 != "pts/2" {print \$1, \$3}' - фильтрует вывод:

- NR > 1 - пропускает первую строку (заголовок)
- \$2 != "pts/2" - выбирает только процессы, у которых второй столбец (tty) НЕ равен "pts/2"
- {print \$1, \$3} - для отфильтрованных строк выводит только PID (столбец 1) и имя команды (столбец 3)

## Задачи из второй части

На рисунке 1 последовательно выполнены пункты 4-4.5

```
bazaraj@bazara-k53sm:/boot$ uname -a
Linux bazara-k53sm 6.14.0-36-generic #36~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Oct 15 15:45:17 UTC 2 x86_64 x86_64 x86_64 GNU/Linux
bazaraj@bazara-k53sm:/boot$ ps -p $$
    PID TTY      TIME CMD
 15484 pts/2    00:00:00 bash
bazaraj@bazara-k53sm:/boot$ id
uid=1000(bazaraj) gid=1000(bazaraj) группы=1000(bazaraj),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),114(lpadmin),126(libvirt),988(sambashare),993(kvm)
bazaraj@bazara-k53sm:/boot$ whoami
bazaraj
bazaraj@bazara-k53sm:/boot$ free -h
    всего     занят     свободно общая буф/врем. доступно
Память:   7,7Gi   2,0Gi   1,7Gi  251Mi  4,4Gi   5,6Gi
Подкачка: 511Mi   0B   511Mi
bazaraj@bazara-k53sm:/boot$ df -h
Файл.система Размер Использовано Дост Использовано% Смонтировано в
tmpfs    786M   1,9M  784M       1% /run
/dev/sda2   219G   33G  175G      16% /
tmpfs    3,9G   1,1M  3,9G       1% /dev/shm
tmpfs    5,0M   8,0K  5,0M       1% /run/lock
efivarsfs  64K   41K  19K      70% /sys/firmware/efi/efivars
tmpfs    3,9G   0  3,9G       0% /run/qemu
tmpfs    3,9G   12K  3,9G      1% /tmp
/dev/sda1   300M   7,8M  292M      3% /boot/efi
tmpfs    786M   92K  786M      1% /run/user/1000
bazaraj@bazara-k53sm:/boot$
```

Рисунок 4 – пункты 4 – 4.5

На рисунке 2 последовательно выполнены пункты 5 – 5.5.

Информация о выполняющихся процессах текущего пользователя и всех процессов в текущем интерпретаторе команд для удобства чтения записаны в userProcessLR5.txt и allProcessLR5.txt,

ps -u \$USER -o pid,ppid,cmd – для вывода процессов пользователя  
ps aux – для вывода всех процессов

```
- : sudo su — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
root@bazara-k53sm:/home/bazaraj# echo $$
16014
root@bazara-k53sm:/home/bazaraj# echo $PPID
16013
root@bazara-k53sm:/home/bazaraj# ps -p 1 -o pid,comm
  PID COMMAND
    1 systemd
root@bazara-k53sm:/home/bazaraj#
```

Рисунок 5 – пункты 5 – 5.5

На рисунке 3 последовательно выполнены пункты 6 – 6.5.

```
root@bazara-k53sm:/boot# ps -u root -o pid,ppid,nice,cmd | grep -E "(bash/PID)"
15823 15799 0 grep --color=auto -E (bash/PID)
root@bazara-k53sm:/boot# nice
0
root@bazara-k53sm:/boot# nice -n 10 bash
root@bazara-k53sm:/boot# echo $$
15825
root@bazara-k53sm:/boot# [REDACTED]

[REDACTED] sudo su — Konsole <2>
Файл Правка Вид Закладки Модули Настройка Справка
root@bazara-k53sm:/home/bazaraj# renice -n 5 15825
15825 (process ID) старый приоритет 10, новый приоритет 5
root@bazara-k53sm:/home/bazaraj# ps lax | grep bash
0 1000 13402 13387 20 0 14588 8888 do_wai Ss pts/1 0:00 /bin/bash
4 0 13459 13458 20 0 10332 4580 do_wai S pts/2 0:00 bash
4 1000 15484 15483 20 0 14568 8892 do_wai S pts/2 0:00 bash
4 0 15799 15798 20 0 10212 4460 do_wai S pts/3 0:00 bash
0 0 15825 15799 25 5 10212 4496 do_sel SN+ pts/3 0:00 bash
0 1000 15846 15835 20 0 14588 8912 do_wai Ss pts/4 0:00 /bin/bash
4 0 15887 15886 20 0 10212 4500 do_wai S pts/5 0:00 bash
4 0 15898 15887 20 0 9176 2408 pipe_r S+ pts/5 0:00 grep --color=auto bash
root@bazara-k53sm:/home/bazaraj# [REDACTED]
```

Рисунок 6 – пункты 6 – 6.5

Второй терминала запущен для управления приоритетом другого терминала

## Вывод

В ходе выполнения лабораторной работы были успешно приобретены практические навыки работы с процессами в операционной системе Linux. Были изучены основные концепции: идентификация процессов (PID, PPID), их состояния, а также способы управления приоритетами выполнения с помощью команд nice и renice. Получен опыт мониторинга процессов через утилиту ps с различными ключами и фильтрами, что позволяет эффективно анализировать активность в системе.

Освоенные приёмы являются фундаментальными для администрирования Linux-систем, диагностики проблем и оптимизации потребления ресурсов. Цель работы достигнута.

## Контрольные вопросы

1. Перечислите состояния задачи в ОС Ubuntu.

- R (Running/Runnable) — выполняется или готов к выполнению (в очереди планировщика).
- S (Sleeping) — прерываемый сон (ожидание события: ввода/вывода, сигнала и т.д.).
- D (Uninterruptible Sleep) — непрерываемый сон (обычно ожидание завершения операций ввода/вывода низкого уровня). Процесс нельзя убить даже сигналом SIGKILL до завершения операции.
- Z (Zombie) — процесс-зомби. Завершился, но его запись ещё существует в таблице процессов, так как родительский процесс не прочитал его статус завершения (wait()).
- T (Stopped) — процесс остановлен сигналом (например, SIGSTOP, SIGTSTP) или находится под отладчиком.
- t (Tracing stop) — остановлен трассировщиком (тот же T, но для отладки, например, ptrace).
- X (Dead) — процесс полностью завершён (встречается редко, так как состояние очень кратковременное).

2. Как создаются задачи в ОС Ubuntu?

Задачи (процессы) создаются следующими способами:

- Запуск программы из командной строки (в оболочке): например, firefox & или ./script.sh.
- Запуск из графического интерфейса (через меню приложений, ярлыки).
- Системными демонами/службами (через systemd): например, sudo systemctl start nginx.
- Программно через системные вызовы:
  - fork() — создание дочернего процесса-копии родительского.
  - exec() — замещение образа процесса новым исполняемым файлом.
  - clone() — создание нового процесса или потока с более тонким контролем.
- Планировщиком заданий (cron, at) — по расписанию.

3. Назовите классы потоков ОС Ubuntu.

- SCHED\_OTHER (стандартное планирование с разделением времени) — обычные процессы с динамическим приоритетом (используется алгоритм CFS — Completely Fair Scheduler). Управляется значением nice.
- SCHED\_FIFO (планирование "первым пришёл — первым обслужен") — реального времени. Процесс выполняется, пока не освободит CPU добровольно, не будет заблокирован или не будет вытеснен процессом с более высоким приоритетом SCHED\_FIFO.

- **\*\*SCHED\_RR** (циклическое планирование) — реального времени с квантованием времени. Аналогично SCHED\_FIFO, но процессам выделяется фиксированный квант времени.
- **SCHED\_BATCH** — для пакетных задач, похож на SCHED\_OTHER, но оптимизирован для снижения интерактивности (меньше перепланирований).
- **SCHED\_IDLE** — для задач с очень низким приоритетом (ниже, чем nice=19). Выполняются, когда система простоявает.
- **SCHED\_DEADLINE** — самое современное планирование реального времени, где каждый процесс имеет дедлайн для завершения работы.

4. Как используется приоритет планирования при запуске задачи?

- Для обычных процессов (SCHED\_OTHER) используется динамический приоритет, который вычисляется планировщиком CFS на основе:
  - Значения nice (от -20 до 19). Чем ниже значение, тем выше приоритет (больше квантов времени).
  - Истории выполнения процесса (учитывается, сколько CPU процесс уже использовал).
- При запуске задачи можно указать значение nice:
  - nice -n <значение> <команда> — запуск команды с заданным значением nice.
  - По умолчанию процессы наследуют значение nice родителя (обычно 0).
- Для процессов реального времени (SCHED\_FIFO, SCHED\_RR) приоритет задаётся явно числовым значением (от 1 до 99, где 99 — наивысший). Устанавливается через sched\_setscheduler() или команду chrt.

5. Как можно изменить приоритет для выполняющейся задачи?

- Для обычных процессов (SCHED\_OTHER):
  - Команда renice: renice -n <новое\_значение\_nice> -p <PID>.
  - Через оболочку: renice -n 5 1234 (установить nice=5 для процесса с PID=1234).
  - Можно менять приоритет для всех процессов пользователя: renice -n 5 -u username.
- Для процессов реального времени:
  - Команда chrt: chrt -p <приоритет> <PID> или chrt -r -p <приоритет> <PID> (для SCHED\_RR).
  - Например: chrt -r -p 90 1234.
- Программно:
  - Системный вызов setpriority() (для изменения nice).
  - Системный вызов sched\_setscheduler() (для смены класса и приоритета планирования).
- Через файловую систему /proc:

- Для изменения nice: echo <значение> > /proc/<PID>/niceness (редко используется).
- Или через cat /proc/<PID>/stat для просмотра текущего приоритета.

