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 АС-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: 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

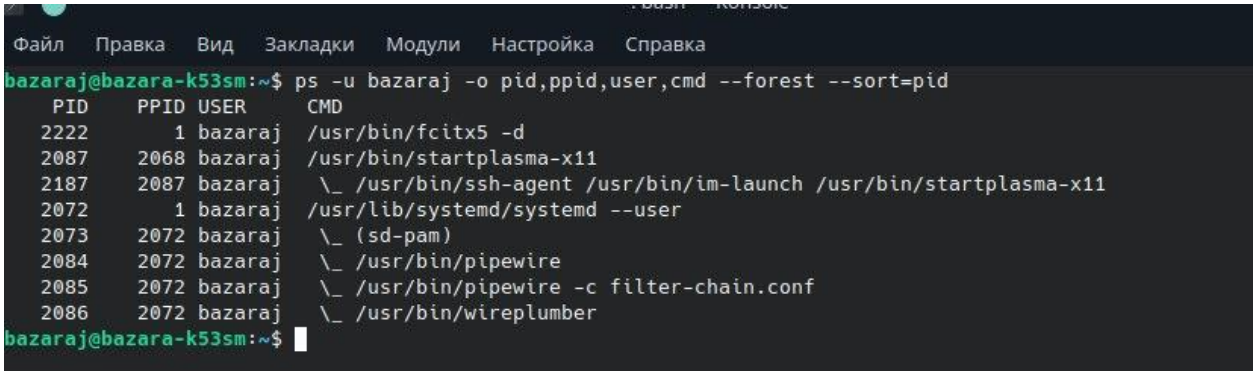

Рисунок 1 – Задание 1

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

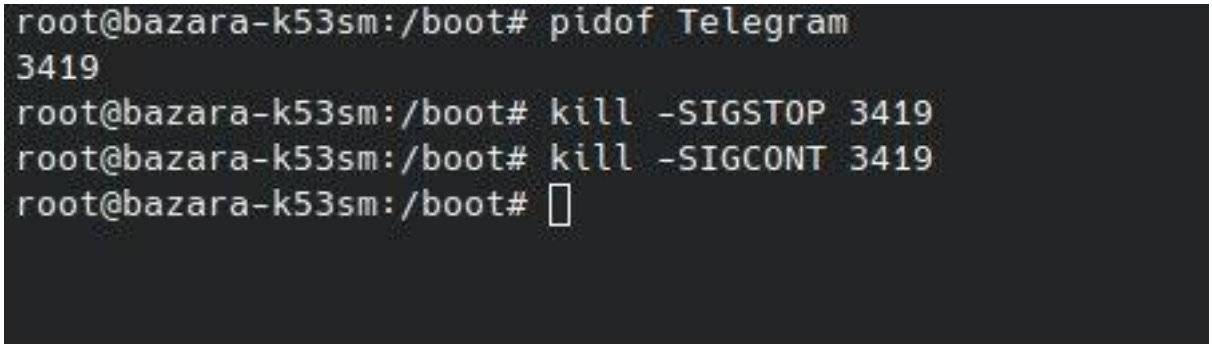

Рисунок 2 – Задание 2

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 | avk ‘NR > 1 && $2!=”pts/2” {print $1, $3}’

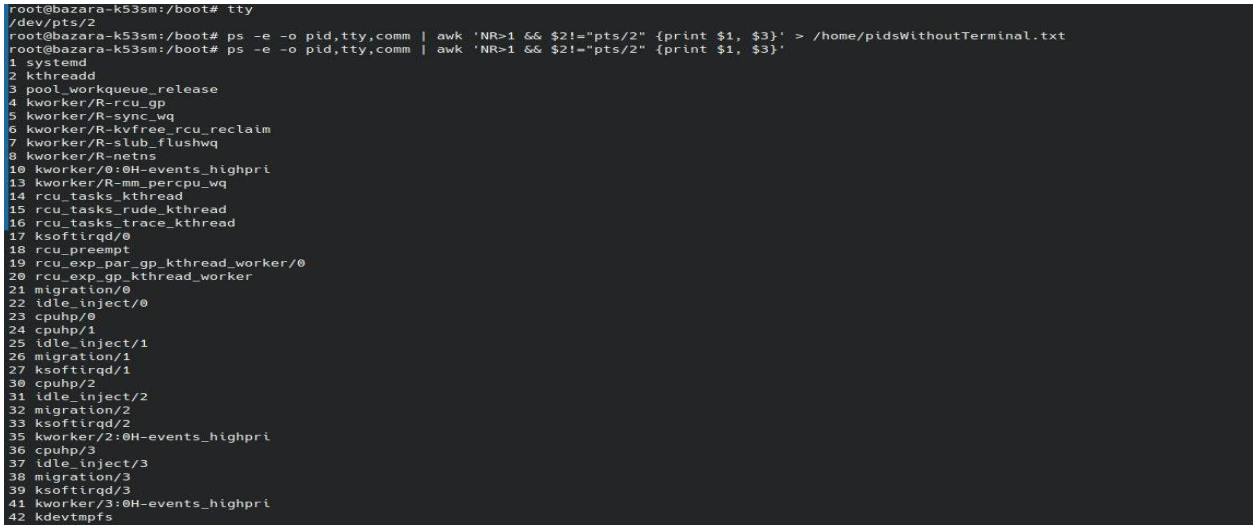

Рисунок 3 – Задание 3

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

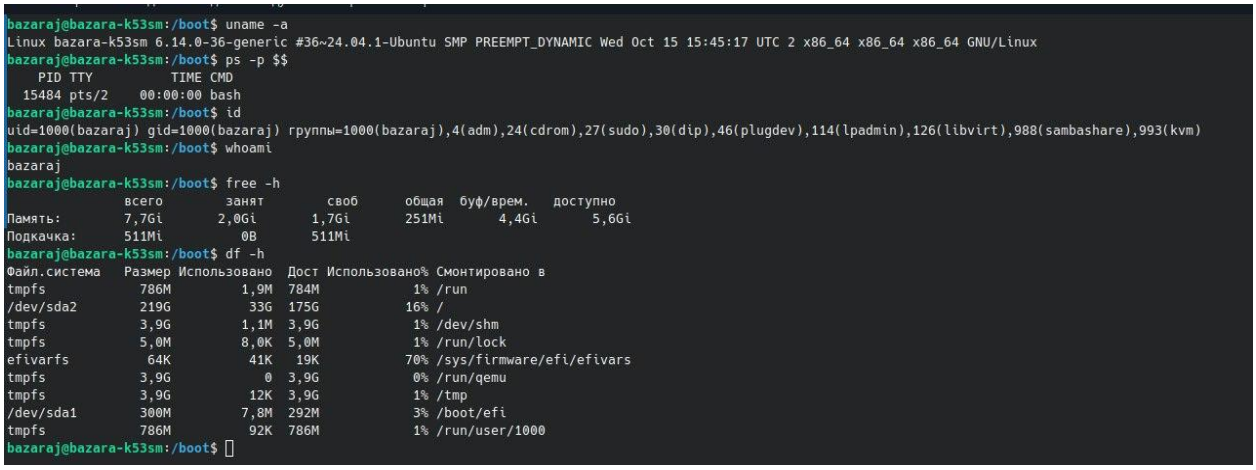

Рисунок 4 – пункты 4 – 4.5

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

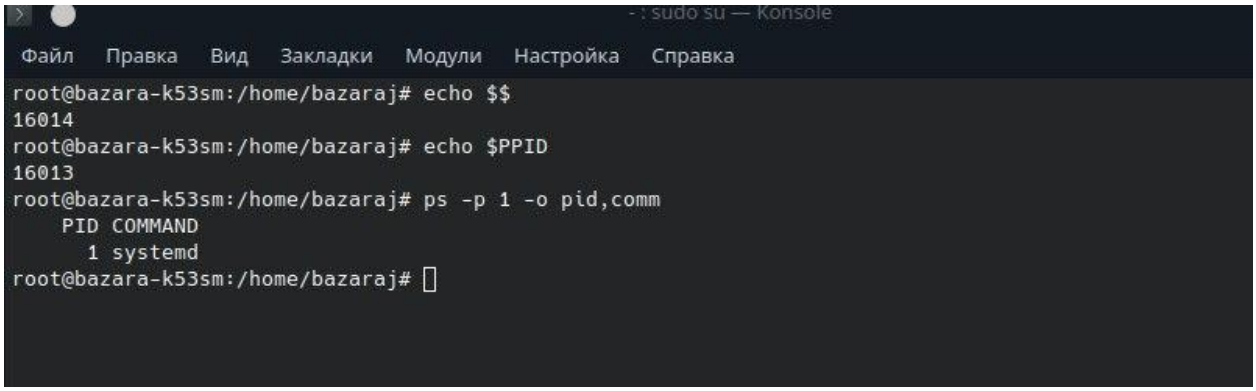

Рисунок 5 – пункты 5 – 5.5

На рисунке 3 последовательно выполнены пункты 6 – 6.5.

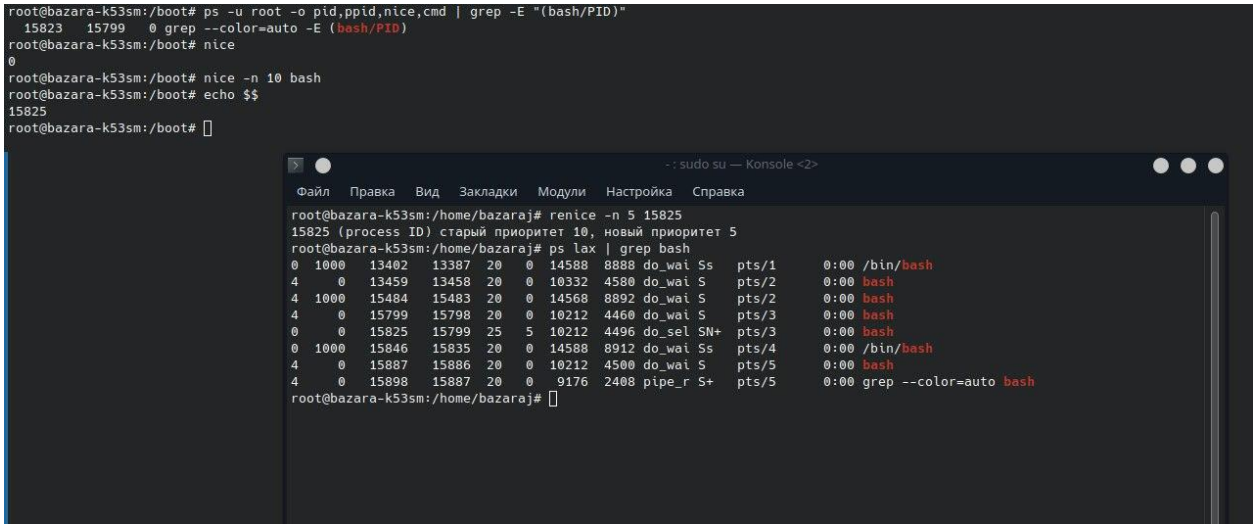

Рисунок 6 – пункты 6 – 6.5

Второй термина 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 простаивает.
- 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.