Леонов Дмитрий

Модуль 1
Практика 3
Раздел 1

1. Войдите под пользователем user1 из практики 2 (su - user1)

```
eltex-pg1-v20@eltex-2025-autumn-13:17> ssh root@172.16.9.194
root@172.16.9.194's password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-87-generic x86_64)
root@eltex-practice2-pg1-v20:~# sudo -u user1 -i
user1@eltex-practice2-pg1-v20:~$ whoami
user1
user1@eltex-practice2-pg1-v20:~$
```

2. Подсчитайте количество процессов, имеющих несколько потоков выполнения
ps -eLf — показать все процессы + потоки (поле NLWP = 5-е по счёту),
awk '$5 > 1' — выбрать только те, у кого потоков > 1,
{count++} END {print count+0} — посчитать и вывести число (если 0 — выведет 0, а не пусто).

```
user1@eltex-practice2-pg1-v20:~$ ps -eLf | awk '$5 > 1 {count++} END {print coun
t+0}'
1
user1@eltex-practice2-pg1-v20:~$
```

3. Запустите top и настройте вывод полей с информацией о процессе следующим образом:

- удалите поля VIRT, RES, SHR;
- добавьте поле RUSER и сделайте так, чтобы это поле было показано после поля USER;

Нажми f — откроется меню полей. Найди VIRT, нажми d — удали.

```
* PID      = Process  SID
* USER     = Effecti  nTH
* PR       = Priorit  P
* NI       = Nice Va  TIME
  VIRT     = Virtual  SWAP
  RES      = Residen  CODE
  SHR      = Shared   DATA
* S        = Process  nMaj
* %CPU     = CPU Usa  nMin
* %MEM     = Memory   nDRT
* TIME+    = CPU Tim  WCHAN
* COMMAND  = Command  Flags
  PPID     = Parent   CGROUPS
  UID      = Effecti  SUPGIDS
  RUID     = Real Us  SUPGRPS
* RUSER    = Real Us  TGID
  SUID     = Saved U  OOMa
```

```
    PID USER      PR  NI S  %CPU  %MEM     TIME+ COMMAND                   RUSER
```

Для того чтобы переместить поле, нажмите на нём **Enter**, а затем стрелку вправо. После этого поле будет захвачено и вы сможете двигать его вверх или вниз. Затем надо отпустить поле на нужном месте с помощью клавиши **Enter**. После завершения настройки можно вернуться обратно в интерфейс программы с помощью клавиши **q**.

```
    PID USER      RUSER     PR  NI S  %CPU  %MEM     TIME+ COMMAND
  29464 user1     user1     20   0 S   0.0   0.1   0:00.01 bash
  29620 user1     user1     20   0 R   0.0   0.2   0:00.15 top
    675 systemd+ systemd+   20   0 S   0.0   0.2   0:01.79 systemd-timesyn
    674 systemd+ systemd+   20   0 S   0.0   0.3   0:00.81 systemd-resolve
```

4. В другом терминальном окне выполните команду passwd и оставьте ее в состоянии запроса текущего пароля

5. Перейдите в терминальное окно с top и выполните следующие действия:

• выведите все процессы, для которых реальным пользователем является пользователь, которым вы вошли в сеанс;

• найдите процесс, запущенный командой passwd;

• отправьте этому процессу сигналы 15 (SIGTERM), 2 (SIGINT), 3 (SIGQUIT), 9(SIGKILL)

Нажми O (фильтр).

```
add filter #2 (ignoring case) as: [!]FLD?VAL RUSER=user1
    PID USER      RUSER     PR  NI S  %CPU  %MEM     TIME+ COMMAND
```

```
    PID USER      RUSER     PR  NI S  %CPU  %MEM     TIME+ COMMAND
  29717 root      user1     20   0 S   0.0   0.1   0:00.00 passwd
  29718 user1     user1     20   0 R   0.0   0.2   0:00.08 top
```

```
user1@eltex-practice2-pg1-v20:~$ kill -15 29717
```

29717 - pid

```
user1@eltex-practice2-pg1-v20:~$ kill -2 29717
user1@eltex-practice2-pg1-v20:~$ kill -3 29717
user1@eltex-practice2-pg1-v20:~$ kill -9 29717
# SIGTERM — попытка мягко завершить
# SIGINT — как Ctrl+C
# SIGQUIT — завершить и сделать core dump
# SIGKILL — принудительно убить
    PID USER      RUSER     PR  NI S  %CPU  %MEM     TIME+ COMMAND
  29707 user1     user1     20   0 S   0.0   0.1   0:00.01 bash
  29738 user1     user1     20   0 R   0.0   0.1   0:00.03 top
```

Процесс завершился

6. Выполните команду vim ~/file_task3.txt и нажмите Ctrl-Z

```
user1@eltex-practice2-pg1-v20:~$ vim ~/file_task3.txt

[1]+  Stopped                 vim ~/file_task3.txt
user1@eltex-practice2-pg1-v20:~$
```

7. Выполните команду sleep 600, нажмите Ctrl-Z и выполните команду jobs

Вывод списка заданий

```
user1@eltex-practice2-pg1-v20:~$ sleep 600
^Z
[2]+  Stopped                 sleep 600
user1@eltex-practice2-pg1-v20:~$ jobs
[1]-  Stopped                 vim ~/file_task3.txt
[2]+  Stopped                 sleep 600
user1@eltex-practice2-pg1-v20:~$
```

8. Последнее задание (sleep 600) сделайте фоновым

```
user1@eltex-practice2-pg1-v20:~$ bg %2
[2]+ sleep 600 &
user1@eltex-practice2-pg1-v20:~$ jobs
[1]+  Stopped                 vim ~/file_task3.txt
[2]-  Running                 sleep 600 &
user1@eltex-practice2-pg1-v20:~$
```

9. Измените число NICE у задания (sleep 600), сделав его равным 10

Найдем pid

```
user1@eltex-practice2-pg1-v20:~$ ps aux | grep sleep
user1      29752  0.0  0.0   5684  2048 pts/1    S    07:10   0:00 sleep 600
```

Установим приоритет = 10

```
user1@eltex-practice2-pg1-v20:~$ renice 10 -p 29752
29752 (process ID) old priority 0, new priority 10
user1@eltex-practice2-pg1-v20:~$
```

10. Проверьте, что число NICE у этого задания изменилось

```
user1@eltex-practice2-pg1-v20:~$ ps -o pid,nice,cmd -p 29752
    PID  NI CMD
  29752  10 sleep 600
```

11. Сделайте задание vim ~/file_ task3.txt активным и выйдите из редактора

```
user1@eltex-practice2-pg1-v20:~$ fg %1
vim ~/file task3.txt
```

Выходим так, завершаем задание :q! + enter

```
~
:q!
```

12. Отправьте сигнал 15 (SIGTERM) заданию sleep 600 и выполните команду jobs

```
user1@eltex-practice2-pg1-v20:~$ ps aux | grep sleep
user1      29771  0.0  0.0   5684  2048 pts/1    T    07:23   0:00 sleep 600
user1@eltex-practice2-pg1-v20:~$ kill -15 29771
user1@eltex-practice2-pg1-v20:~$ jobs
[1]+  Stopped                 sleep 600
```

13. Создайте перехватчик сигналов SIGINT и SIGQUIT внутри командного интерпретатора, который выводит сообщение «Меня голыми руками не возьмёшь!» (используйте встроенную команду trap) и отправьте сигналы самому себе

```
user1@eltex-practice2-pg1-v20:~$ trap 'echo "Меня голыми руками не возьмёшь!"' SI
GINT SIGQUIT
user1@eltex-practice2-pg1-v20:~$ kill -SIGINT $$
Меня голыми руками не возьмёшь!
```

Раздел 2

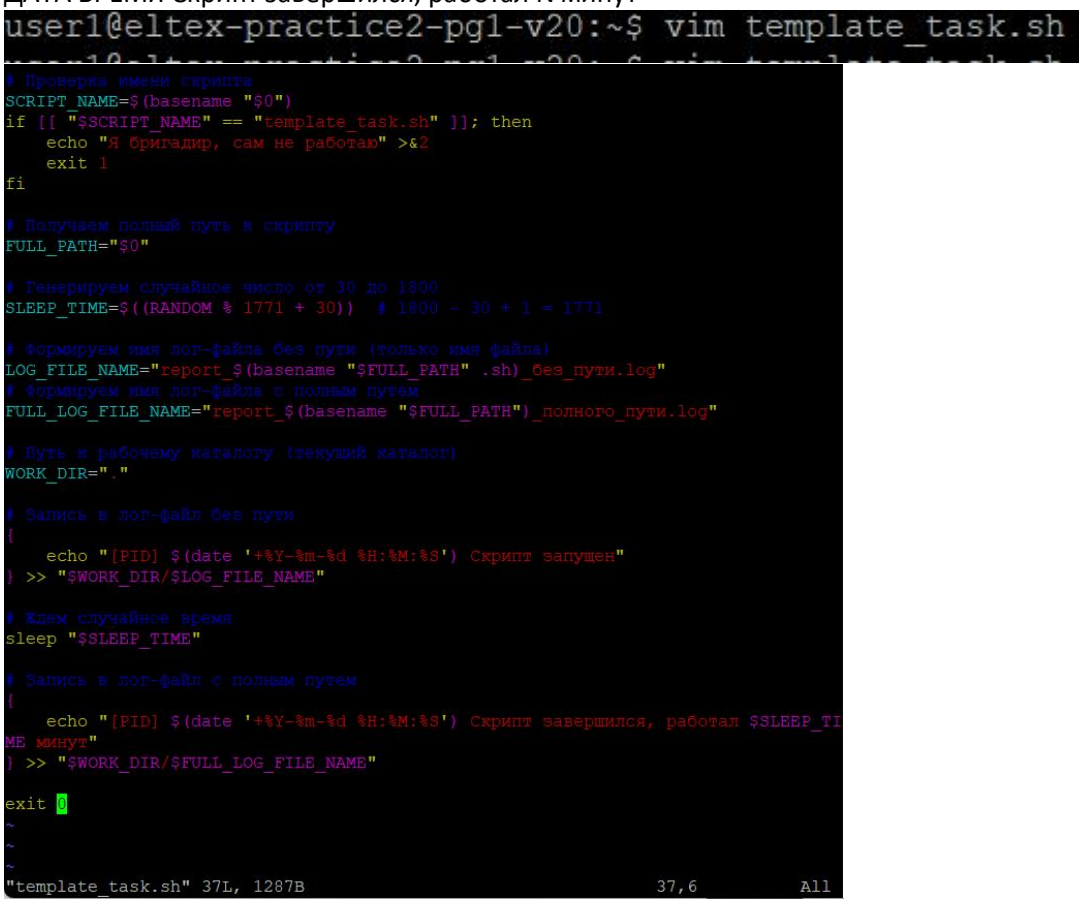
1. Создайте скрипт на языке bash с именем template_task.sh, делающий следующее:

При запуске проверят, что имя скрипта не совпадает с template_task.sh, если совпадает - выходит с уведомлением «я бригадир, сам не работаю»

При запуске дописывает в файл report_имя_скрипта_без_полного_пути.log в рабочем каталоге информацию: [PID] ДАТА ВРЕМЯ Скрипт запущен

Генерирует случайное число от 30 до 1800 и ждет такое количество секунд

Дописывает в файл report_имя_скрипта_без_полного_пути.log сообщение: [PID] ДАТА ВРЕМЯ Скрипт завершился, работал N минут



Выход - :wq + enter

Файл template_task.sh:

```
#!/bin/bash

# Проверка имени скрипта
SCRIPT_NAME=$(basename "$0")
if [[ "$SCRIPT_NAME" == "template_task.sh" ]]; then
   echo "Я бригадир, сам не работаю" >&2
fi

# Получаем полный путь к скрипту
FULL_PATH="$0"

# Генерируем случайное число от 30 до 1800
SLEEP_TIME=$((RANDOM % 1771 + 30))  # 1800 - 30 + 1 = 1771

# Формируем имя лог-файла без пути (только имя файла)
```

```
LOG_FILE_NAME="report_$(basename "$FULL_PATH")_без_полного_пути.log"

# Путь к рабочему каталогу (текущий каталог)
WORK_DIR="."

# Запись в лог-файл без пути
{
  echo "[PID $$] $(date '+%Y-%m-%d %H:%M:%S') Скрипт запущен"
} > "$WORK_DIR/$LOG_FILE_NAME"

echo "Sleep начался. Будет работать $((SLEEP_TIME/60)) минут ($(($SLEEP_TIME%60)) сек)"
# Ждем случайное время
sleep "$SLEEP_TIME"

# Запись в лог-файл с полным путем
{
  echo "[PID $$] $(date '+%Y-%m-%d %H:%M:%S') Скрипт завершился, работал $SLEEP_TIME минут"
} >> "$WORK_DIR/$LOG_FILE_NAME"

exit 0
```

```
user1@eltex-practice2-pg1-v20:~$ ./template_task.sh
Я бригадир, сам не работаю
Sleep начался. Будет работать 0 минут (39 сек)
```

```
user1@eltex-practice2-pg1-v20:~$ ls
report_template_task.sh_без_полного_пути.log  template_task.sh
user1@eltex-practice2-pg1-v20:~$ cat report_template_task.sh_без_полного_пути.log
[PID 30067] 2025-11-21 10:03:57 Скрипт запущен
[PID 30067] 2025-11-21 10:04:36 Скрипт завершился, работал 39 минут
user1@eltex-practice2-pg1-v20:~$
```

2. Создайте скрипт на языке bash с именем observer.sh, читающий файл конфигурации со списком скриптов observer.conf, проверяющим их наличие в списке работающих процессов поиском в /proc и запускающих их в отключенном от терминала режиме (nohup) в случае отсутствия в нем. Информация о перезапуске дописывайте в файл observer.log

Файл observer.sh:

```
#!/bin/bash

CONFIG_FILE="observer.conf"
LOG_FILE="observer.log"

# Проверяем существование конфигурационного файла
[[ -f "$CONFIG_FILE" ]] || {
  echo "Ошибка: файл конфигурации '$CONFIG_FILE' не найден." >&2
  exit 1
}

# Функция: проверяет, запущен ли скрипт по его полному или относительному пути
is_running_via_proc() {
  local target="$1"
```

```
    local abs_target

    # Приводим к абсолютному пути для корректного сравнения
    if [[ "$target" == /* ]]; then
        abs_target="$target"
    else
        abs_target="$PWD/$target"
    fi

    # Обходим все PID-директории в /proc
    for pid_dir in /proc/[0-9]*; do
        [[ -d "$pid_dir" ]] || continue

        cmdline_file="$pid_dir/cmdline"
        [[ -r "$cmdline_file" ]] || continue

        # Читаем cmdline и заменяем \0 на пробелы
        cmdline=$(tr '\0' ' ' < "$cmdline_file" 2>/dev/null)

        # Пропускаем пустые или некорректные записи
        [[ -z "$cmdline" ]] && continue

        # Извлекаем первый аргумент (путь к запущенной программе)
        first_arg=$(echo "$cmdline" | awk '{print $1}')

        # Проверяем совпадение с абсолютным путём
        if [[ "$first_arg" == "$abs_target" ]] || [[ "$first_arg" == "$target" ]]; then
            return 0  # найден запущенный процесс
        fi
    done

    return 1  # не найден
}

# Основной цикл
while IFS= read -r script_name; do
    [[ -z "$script_name" || "$script_name" =~ ^[[:space:]]*# ]] && continue

    # Проверка — ищем в /proc
    if ! is_running_via_proc "$script_name"; then
        # Запускаем через nohup
        {
            echo "$(date '+%Y-%m-%d %H:%M:%S') Перезапущен скрипт: $script_name"
        } >> "$LOG_FILE"
        nohup "$script_name" > /dev/null 2>&1 &
    fi
done < "$CONFIG_FILE"

exit 0
user1@eltex-practice2-pg1-v20:~$ chmod +x observer.sh
```

```
user1@eltex-practice2-pg1-v20:~$ touch observer.conf
user1@eltex-practice2-pg1-v20:~$
user1@eltex-practice2-pg1-v20:~$ echo "./template_task.sh" > observer.conf
user1@eltex-practice2-pg1-v20:~$ ./observer.sh
user1@eltex-practice2-pg1-v20:~$ cat observer.log
Запускаем скрипт: ./template_task.sh
2025-11-21 10:13:11 Перезапущен скрипт: ./template_task.sh
```

3. Настройте запуск observer.sh посредством cron по расписанию – 1 раз в минуту

```
user1@eltex-practice2-pg1-v20:~$ crontab -e
no crontab for user1 - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed
Choose 1-4 [1]: 1
```

```
user1@eltex-practice2-pg1-v20: ~
  GNU nano 7.2                  /tmp/crontab.kJ36ro/crontab *
* * * * * /home/user1/observer.sh > /dev/null 2>&1
# Edit this file to introduce tasks to be run by cron.
#
File Name to Write: /tmp/crontab.kJ36ro/crontab
^G Help          M-D DOS Format      M-A Append          M-B Backup File
^C Cancel        M-M Mac Format      M-P Prepend         ^T Browse
```

Enter

```
user1@eltex-practice2-pg1-v20:~$ crontab -e
no crontab for user1 - using an empty one
crontab: installing new crontab
```

4. Создайте несколько символьных ссылок на файл template_task.sh с различными именами (рабочие задачи), добавьте в файл конфигурации observer.conf соответствующие записи об этих задачах, включая исходный файл template_task.sh

```
user1@eltex-practice2-pg1-v20:~$ ln -s template_task.sh task1.sh
ln -s template_task.sh task2.sh
ln -s template_task.sh task3.sh
user1@eltex-practice2-pg1-v20:~$ echo "./task1.sh" >> observer.conf
echo "./task2.sh" >> observer.conf
echo "./task3.sh" >> observer.conf
user1@eltex-practice2-pg1-v20:~$ cat observer.conf
./template_task.sh
./task1.sh
./task2.sh
./task3.sh
user1@eltex-practice2-pg1-v20:~$
user1@eltex-practice2-pg1-v20:~$ chmod +x task1.sh task2.sh task3.sh
```

5. Соберите статистику работы в виде набора файлов report_*.log, observer.log, приложите их вместе с исходными текстами скриптов в качестве отчета в виде сжатого архива tar. Не забудьте остановить процесс, удалив задачу в cron!

```
user1@eltex-practice2-pg1-v20:~$ crontab -e
crontab: installing new crontab
user1@eltex-practice2-pg1-v20:~$ crontab -r
```

```
user1@eltex-practice2-pg1-v20:~$ tar -czf report_$(date +%Y%m%d_%H%M%S).tar.gz \
    report_*.log \
    observer.log \
    template_task.sh \
    observer.sh \
    observer.conf \
    task1.sh \
    task2.sh \
    task3.sh
user1@eltex-practice2-pg1-v20:~$ tar -tzf report_*.tar.gz
report_template_task.sh_без_полного_пути.log
observer.log
template_task.sh
observer.sh
observer.conf
task1.sh
task2.sh
task3.sh
```