Практика 3

Раздел 1

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

```
root@eltex-practice2-pg1-v6:~# su - user1
user1@eltex-practice2-pg1-v6:~$
```

2.Подсчитываем количество процессов, имеющих несколько потоков выполнения

```
user1@eltex-practice2-pg1-v6:~$ ps -eLo pid | sort | uniq -c | awk '$1 > 1' | wc
 -l
10
user1@eltex-practice2-pg1-v6:~$
```

3.Запускаем top и настраиваем вывод полей с информацией о процессе следующим образом:

• удаляем поля VIRT, RES, SHR;

• добавляем поле RUSER и делаем так, чтобы это поле было показано после поля USER;

```
top - 06:10:17 up 3 days,  3:49,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 110 total,   1 running, 109 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 65.0 id, 35.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3916.0 total,   2265.6 free,    529.2 used,   1421.2 buff/cache
MiB Swap:   3185.0 total,   3184.7 free,      0.3 used.   3386.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	22544	13824	9600	S	0.0	0.3	0:07.06	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_wo+
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
16	root	20	0	0	0	0	S	0.0	0.0	0:00.10	ksoftir+
17	root	20	0	0	0	0	I	0.0	0.0	0:01.92	rcu_pre+
18	root	rt	0	0	0	0	S	0.0	0.0	0:01.06	migrati+
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_in+
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0

Нажимаем f для выбора поля

```
Fields Management for window 1:Def, whose current sort field is %CPU
   Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
   'd' or <Space> toggles display, 's' sets sort.  Use 'q' or <Esc> to end!

* PID      = Process  PGRP     = Process  OOMs     = OOMEM S  RSS      = Res Mem
* USER     = Effecti  TTY      = Control  ENVIRON  = Environ  PSS      = Proport
* PR       = Priorit  TPGID    = Tty Pro  vMj      = Major F  PSan     = Proport
* NI       = Nice Va  SID      = Session  vMn      = Minor F  PSfd     = Proport
* VIRT     = Virtual  nTH      = Number   USED     = Res+Swa  PSsh     = Proport
* RES      = Residen  P        = Last Us  nsIPC    = IPC nam  USS      = Unique
* SHR      = Shared   TIME     = CPU Tim  nsMNT    = MNT nam  ioR      = I/O Byt
* S        = Process  SWAP     = Swapped  nsNET    = NET nam  ioRop    = I/O Rea
* %CPU     = CPU Usa  CODE     = Code Si  nsPID    = PID nam  ioW      = I/O Byt
* %MEM     = Memory   DATA     = Data+St  nsUSER   = USER na  ioWop    = I/O Wri
* TIME+    = CPU Tim  nMaj     = Major P  nsUTS    = UTS nam  AGID     = Autogro
* COMMAND  = Command  nMin     = Minor P  LXC      = LXC con  AGNI     = Autogro
  PPID     = Parent   nDRT     = Dirty P  RSan     = RES Ano  STARTED  = Start T
  UID      = Effecti  WCHAN    = Sleepin  RSfd     = RES Fil  ELAPSED  = Elapsed
  RUID     = Real Us  Flags    = Task Fl  RSlk     = RES Loc  %CUU     = CPU Uti
  RUSER    = Real Us  CGROUPS  = Control  RSsh     = RES Sha  %CUC     = Utiliza
  SUID     = Saved U  SUPGIDS  = Supp Gr  CGNAME   = Control  nsCGROUP = CGRP na
  SUSER    = Saved U  SUPGRPS  = Supp Gr  NU       = Last Us  nsTIME   = TIME na
  GID      = Group I  TGID     = Thread   LOGID    = Login U
  GROUP    = Group N  OOMa     = OOMEM A  EXE      = Executa
```

Отключаем, нажимая d

```
Fields Management for window 1:Def, whose current sort field is %CPU
   Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
   'd' or <Space> toggles display, 's' sets sort.  Use 'q' or <Esc> to end!

* PID      = Process Id             PGRP     = Process Group Id      OOMs     = OOMEM Score current   RSS      = Res Mem (smaps), KiB
* USER     = Effective User Name    TTY      = Controlling Tty       ENVIRON  = Environment vars      PSS      = Proportion RSS,  KiB
* PR       = Priority               TPGID    = Tty Process Grp Id    vMj      = Major Faults delta    PSan     = Proportion Anon, KiB
* NI       = Nice Value             SID      = Session Id            vMn      = Minor Faults delta    PSfd     = Proportion File, KiB
  VIRT     = Virtual Image (KiB)    nTH      = Number of Threads     USED     = Res+Swap Size (KiB)   PSsh     = Proportion Shrd, KiB
  RES      = Resident Size (KiB)    P        = Last Used Cpu (SMP)   nsIPC    = IPC namespace Inode   USS      = Unique RSS, KiB
  SHR      = Shared Memory (KiB)    TIME     = CPU Time              nsMNT    = MNT namespace Inode   ioR      = I/O Bytes Read
* S        = Process Status         SWAP     = Swapped Size (KiB)    nsNET    = NET namespace Inode   ioRop    = I/O Read Operations
* %CPU     = CPU Usage              CODE     = Code Size (KiB)       nsPID    = PID namespace Inode   ioW      = I/O Bytes Written
* %MEM     = Memory Usage (RES)     DATA     = Data+Stack (KiB)      nsUSER   = USER namespace Inode  ioWop    = I/O Write Operations
* TIME+    = CPU Time, hundredths   nMaj     = Major Page Faults     nsUTS    = UTS namespace Inode   AGID     = Autogroup Identifier
* COMMAND  = Command Name/Line      nMin     = Minor Page Faults     LXC      = LXC container name    AGNI     = Autogroup Nice Value
  PPID     = Parent Process pid     nDRT     = Dirty Pages Count     RSan     = RES Anonymous (KiB)   STARTED  = Start Time from boot
  UID      = Effective User Id      WCHAN    = Sleeping in Function  RSfd     = RES File-based (KiB)  ELAPSED  = Elapsed Running Time
  RUID     = Real User Id           Flags    = Task Flags <sched.h>  RSlk     = RES Locked (KiB)      %CUU     = CPU Utilization
  RUSER    = Real User Name         CGROUPS  = Control Groups        RSsh     = RES Shared (KiB)      %CUC     = Utilization + child
  SUID     = Saved User Id          SUPGIDS  = Supp Groups IDs       CGNAME   = Control Group name    nsCGROUP = CGRP namespace Inode
  SUSER    = Saved User Name        SUPGRPS  = Supp Groups Names     NU       = Last Used NUMA node   nsTIME   = TIME namespace Inode
  GID      = Group Id               TGID     = Thread Group Id       LOGID    = Login User Id
  GROUP    = Group Name             OOMa     = OOMEM Adjustment      EXE      = Executable Path
```

Включаем, нажимая d

```
Fields Management for window 1:Def, whose current sort field is %CPU
   Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
   'd' or <Space> toggles display, 's' sets sort.  Use 'q' or <Esc> to end!

* PID      = Process Id             PGRP     = Process Group Id      OOMs     = OOMEM Score current   RSS      = Res Mem (smaps), KiB
* USER     = Effective User Name    TTY      = Controlling Tty       ENVIRON  = Environment vars      PSS      = Proportion RSS,  KiB
* PR       = Priority               TPGID    = Tty Process Grp Id    vMj      = Major Faults delta    PSan     = Proportion Anon, KiB
* NI       = Nice Value             SID      = Session Id            vMn      = Minor Faults delta    PSfd     = Proportion File, KiB
  VIRT     = Virtual Image (KiB)    nTH      = Number of Threads     USED     = Res+Swap Size (KiB)   PSsh     = Proportion Shrd, KiB
  RES      = Resident Size (KiB)    P        = Last Used Cpu (SMP)   nsIPC    = IPC namespace Inode   USS      = Unique RSS, KiB
  SHR      = Shared Memory (KiB)    TIME     = CPU Time              nsMNT    = MNT namespace Inode   ioR      = I/O Bytes Read
* S        = Process Status         SWAP     = Swapped Size (KiB)    nsNET    = NET namespace Inode   ioRop    = I/O Read Operations
* %CPU     = CPU Usage              CODE     = Code Size (KiB)       nsPID    = PID namespace Inode   ioW      = I/O Bytes Written
* %MEM     = Memory Usage (RES)     DATA     = Data+Stack (KiB)      nsUSER   = USER namespace Inode  ioWop    = I/O Write Operations
* TIME+    = CPU Time, hundredths   nMaj     = Major Page Faults     nsUTS    = UTS namespace Inode   AGID     = Autogroup Identifier
* COMMAND  = Command Name/Line      nMin     = Minor Page Faults     LXC      = LXC container name    AGNI     = Autogroup Nice Value
  PPID     = Parent Process pid     nDRT     = Dirty Pages Count     RSan     = RES Anonymous (KiB)   STARTED  = Start Time from boot
  UID      = Effective User Id      WCHAN    = Sleeping in Function  RSfd     = RES File-based (KiB)  ELAPSED  = Elapsed Running Time
  RUID     = Real User Id           Flags    = Task Flags <sched.h>  RSlk     = RES Locked (KiB)      %CUU     = CPU Utilization
* RUSER    = Real User Name         CGROUPS  = Control Groups        RSsh     = RES Shared (KiB)      %CUC     = Utilization + child
  SUID     = Saved User Id          SUPGIDS  = Supp Groups IDs       CGNAME   = Control Group name    nsCGROUP = CGRP namespace Inode
  SUSER    = Saved User Name        SUPGRPS  = Supp Groups Names     NU       = Last Used NUMA node   nsTIME   = TIME namespace Inode
  GID      = Group Id               TGID     = Thread Group Id       LOGID    = Login User Id
  GROUP    = Group Name             OOMa     = OOMEM Adjustment      EXE      = Executable Path
```

Перемещаем с помощью s и стрелок

```
Fields Management for window 4:Usr, whose current sort field is USER
   Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
   'd' or <Space> toggles display, 's' sets sort.  Use 'q' or <Esc> to end!

* PID      = Process I  NI       = Nice Valu  OOMs     = OOMEM Sco  RSS      = Res Mem (
* PPID     = Parent Pr  nTH      = Number of  ENVIRON  = Environme  PSS      = Proportio
* UID      = Effective  P        = Last Used  vMj      = Major Fau  PSan     = Proportio
* USER     = Effective  TIME     = CPU Time   vMn      = Minor Fau  PSfd     = Proportio
* RUSER    = Real User  VIRT     = Virtual I  USED     = Res+Swap   PSsh     = Proportio
* TTY      = Controlli  SWAP     = Swapped S  nsIPC    = IPC names  USS      = Unique RS
* TIME+    = CPU Time,  RES      = Resident   nsMNT    = MNT names  ioR      = I/O Bytes
* %CPU     = CPU Usage  CODE     = Code Size  nsNET    = NET names  ioRop    = I/O Read
* %MEM     = Memory Us  DATA     = Data+Stac  nsPID    = PID names  ioW      = I/O Bytes
* S        = Process S  SHR      = Shared Me  nsUSER   = USER name  ioWop    = I/O Write
* COMMAND  = Command N  nMaj     = Major Pag  nsUTS    = UTS names  AGID     = Autogroup
  RUID     = Real User  nMin     = Minor Pag  LXC      = LXC conta  AGNI     = Autogroup
  SUID     = Saved Use  nDRT     = Dirty Pag  RSan     = RES Anony  STARTED  = Start Tim
  SUSER    = Saved Use  WCHAN    = Sleeping   RSfd     = RES File-  ELAPSED  = Elapsed R
  GID      = Group Id   Flags    = Task Flag  RSlk     = RES Locke  %CUU     = CPU Utili
  GROUP    = Group Nam  CGROUPS  = Control G  RSsh     = RES Share  %CUC     = Utilizati
  PGRP     = Process G  SUPGIDS  = Supp Grou  CGNAME   = Control G  nsCGROUP = CGRP name
  TPGID    = Tty Proce  SUPGRPS  = Supp Grou  NU       = Last Used  nsTIME   = TIME name
  SID      = Session I  TGID     = Thread Gr  LOGID    = Login Use
  PR       = Priority   OOMa     = OOMEM Adj  EXE      = Executabl
```

4.В другом терминальном окне выполняем команду passwd и оставляем ее в состоянии запроса текущего пароля

```
root@eltex-practice2-pg1-v6:~# passwd
New password:
```

5. Перейдем в терминальное окно с top и выполним следующие действия:
• выведем все процессы, для которых реальным пользователем является пользователь, которым мы вошли в сеанс;
• найдем процесс, запущенный командой passwd;
• отправим этому процессу сигналы 15 (SIGTERM), 2 (SIGINT), 3 (SIGQUIT), 9(SIGKILL)

```
top - 06:19:47 up 3 days,  3:59,  3 users,  load average: 0.00, 0.00, 0.00
Tasks: 111 total,   1 running, 110 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3916.0 total,   2261.6 free,    533.2 used,   1421.3 buff/cache
MiB Swap:   3185.0 total,   3184.7 free,      0.3 used.   3382.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  42345 user1     20   0    8644   5376   3840 S   0.0   0.1   0:00.02 bash
  42458 user1     20   0   11936   5888   3712 R   0.0   0.1   0:00.01 top



  42493 root      20   0    9172   3712   3456 S   0.0   0.1   0:00.00 passwd
  42496 user1     20   0   11936   5888   3712 R   0.0   0.1   0:00.26 top
```

```
user1@eltex-practice2-pg1-v6:~$ kill -15 42493
user1@eltex-practice2-pg1-v6:~$ kill -2 42493
user1@eltex-practice2-pg1-v6:~$ kill -3 42493
user1@eltex-practice2-pg1-v6:~$ kill -9 42493
user1@eltex-practice2-pg1-v6:~$
user1@eltex-practice2-pg1-v6:~$ pgrep -a passwd
user1@eltex-practice2-pg1-v6:~$ passwd
Changing password for user1.
Current password: Killed
user1@eltex-practice2-pg1-v6:~$
```

6.Выполним команду vim ~/file_task3.txt и нажмем Ctrl-Z

```
user1@eltex-practice2-pg1-v6:~$ vim ~/file_task3.txt

[1]+  Stopped                 vim ~/file_task3.txt
user1@eltex-practice2-pg1-v6:~$
```

7.Выполним команду sleep 600, нажмем Ctrl-Z и выполним команду jobs

```
user1@eltex-practice2-pg1-v6:~$ sleep 600
^Z
[2]+  Stopped                 sleep 600
user1@eltex-practice2-pg1-v6:~$ jobs
[1]-  Stopped                 vim ~/file_task3.txt
[2]+  Stopped                 sleep 600
user1@eltex-practice2-pg1-v6:~$
```

8.Последнее задание (sleep 600) сделаем фоновым

```
user1@eltex-practice2-pg1-v6:~$ bg %2
[2]+ sleep 600 &
user1@eltex-practice2-pg1-v6:~$
```

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

```
user1@eltex-practice2-pg1-v6:~$ renice -n 10 -p $(pgrep sleep)
42513 (process ID) old priority 0, new priority 10
user1@eltex-practice2-pg1-v6:~$
```

10. Проверим, что число NICE у этого задания изменилось

```
user1@eltex-practice2-pg1-v6:~$ ps -o pid,ni,cmd -C sleep
    PID  NI CMD
  42513  10 sleep 600
user1@eltex-practice2-pg1-v6:~$
```

11.Сделаем задание vim ~/file_ task3.txt активным и выйдем из редактора

```
user1@eltex-practice2-pg1-v6:~$ fg %1
vim ~/file_task3.txt
user1@eltex-practice2-pg1-v6:~$
```

12.Отправим сигнал 15 (SIGTERM) заданию sleep 600 и выполним команду Jobs

```
user1@eltex-practice2-pg1-v6:~$ kill -15 %2
user1@eltex-practice2-pg1-v6:~$ jobs
[2]+  Terminated              sleep 600
user1@eltex-practice2-pg1-v6:~$
```

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

```
user1@eltex-practice2-pg1-v6:~$ trap 'echo "Меня голыми руками не возьмешь!"' SIGINT SIGQUI
T
user1@eltex-practice2-pg1-v6:~$ kill -2 $$
Меня голыми руками не возьмешь!
user1@eltex-practice2-pg1-v6:~$ kill -3 $$
Меня голыми руками не возьмешь!
user1@eltex-practice2-pg1-v6:~$
```

Раздел 2

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

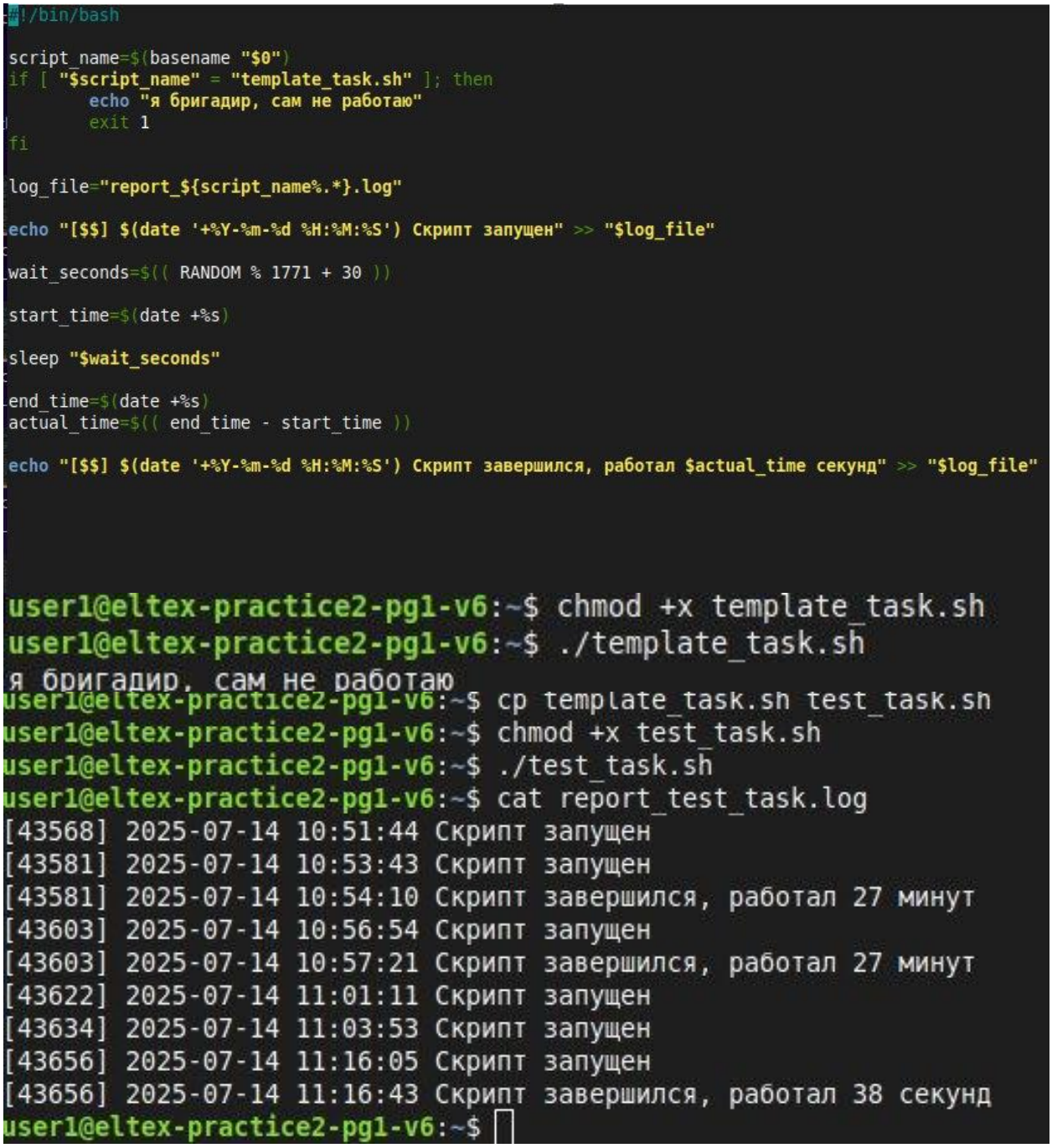
•При запуске проверят, что имя скрипта не совпадает с template_task.sh, если совпадает - выходит с уведомлением «я бригадир, сам не работаю»

•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

```
user1@eltex-practice2-pg1-v6:~$ touch template_task.sh
user1@eltex-practice2-pg1-v6:~$ nano template_task.sh
```



2.Создадим 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 дописываем в файл observer.log

```
user1@eltex-practice2-pg1-v6:~$ touch observer.sh
user1@eltex-practice2-pg1-v6:~$ nano observer.sh
user1@eltex-practice2-pg1-v6:~$ touch observer.conf
user1@eltex-practice2-pg1-v6:~$ nano observer.conf
user1@eltex-practice2-pg1-v6:~$ chmod +x observer.sh
user1@eltex-practice2-pg1-v6:~$ ./observer.sh
user1@eltex-practice2-pg1-v6:~$ cat observer.log
[2025-07-14 11:27:17] Перезапущен скрипт: script1.sh
[2025-07-14 11:27:17] Перезапущен скрипт: script2.sh
[2025-07-14 11:27:17] Перезапущен скрипт: /path/to/script3.sh
user1@eltex-practice2-pg1-v6:~$
```

```
#!/bin/bash

# Файлы конфигурации и логов
CONFIG_FILE="observer.conf"
LOG_FILE="observer.log"

while IFS= read -r script_name || [ -n "$script_name" ]; do
    [ -z "$script_name" ] && continue
    [[ "$script_name" =~ ^# ]] && continue

    pgrep -f "$script_name" >/dev/null 2>&1
    if [ $? -ne 0 ]; then
        nohup bash "$script_name" >/dev/null 2>&1 &

        echo "[$(date '+%Y-%m-%d %H:%M:%S')] Перезапущен скрипт: $script_name" >> "$LOG_FILE"
    fi
done < "$CONFIG_FILE"
```

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

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * /home/user1/observer.sh >>/home/user1/observer.log 2>&1
```

```
user1@eltex-practice2-pg1-v6:~$ tail -f observer.log
[2025-07-14 11:27:17] Перезапущен скрипт: /path/to/script3.sh
[2025-07-14 11:45:01] Перезапущен скрипт: script1.sh
[2025-07-14 11:45:01] Перезапущен скрипт: script2.sh
[2025-07-14 11:45:01] Перезапущен скрипт: /path/to/script3.sh
[2025-07-14 11:46:01] Перезапущен скрипт: script1.sh
[2025-07-14 11:46:01] Перезапущен скрипт: script2.sh
[2025-07-14 11:46:01] Перезапущен скрипт: /path/to/script3.sh
[2025-07-14 11:47:01] Перезапущен скрипт: script1.sh
[2025-07-14 11:47:01] Перезапущен скрипт: script2.sh
[2025-07-14 11:47:01] Перезапущен скрипт: /path/to/script3.sh
```

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

```
user1@eltex-practice2-pg1-v6:~$ ln -s template_task.sh backup_task.sh
user1@eltex-practice2-pg1-v6:~$ ln -s template_task.sh cleanup_task.sh
user1@eltex-practice2-pg1-v6:~$ ln -s template_task.sh report_task.sh
user1@eltex-practice2-pg1-v6:~$ ln -s template_task.sh monitor_task.sh
user1@eltex-practice2-pg1-v6:~$ ls -l *task.sh
lrwxrwxrwx 1 user1 user1  16 Jul 14 11:55 backup_task.sh -> template_task.sh
lrwxrwxrwx 1 user1 user1  16 Jul 14 11:56 cleanup_task.sh -> template_task.sh
lrwxrwxrwx 1 user1 user1  16 Jul 14 11:56 monitor_task.sh -> template_task.sh
lrwxrwxrwx 1 user1 user1  16 Jul 14 11:56 report_task.sh -> template_task.sh
-rwxrwxr-x 1 user1 user1 567 Jul 14 11:18 template_task.sh
-rwxrwxr-x 1 user1 user1 565 Jul 14 11:15 test_task.sh
user1@eltex-practice2-pg1-v6:~$ nano observer.conf
user1@eltex-practice2-pg1-v6:~$ chmod +x *task.sh
user1@eltex-practice2-pg1-v6:~$ ./observer.sh
user1@eltex-practice2-pg1-v6:~$ tail -f observer.log
[2025-07-14 11:56:01] Перезапущен скрипт: /path/to/script3.sh
[2025-07-14 11:57:01] Перезапущен скрипт: script1.sh
[2025-07-14 11:57:01] Перезапущен скрипт: script2.sh
[2025-07-14 11:57:01] Перезапущен скрипт: /path/to/script3.sh
[2025-07-14 11:58:01] Перезапущен скрипт: backup_task.sh
[2025-07-14 11:58:01] Перезапущен скрипт: cleanup_task.sh
[2025-07-14 11:58:01] Перезапущен скрипт: report_task.sh
[2025-07-14 11:58:01] Перезапущен скрипт: monitor_task.sh
[2025-07-14 11:58:01] Перезапущен скрипт: template_task.sh
[2025-07-14 11:58:07] Перезапущен скрипт: template_task.sh
```