

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Операционная система Linux»

Процессы в операционной системе Linux

Студентка

Жидков И.А.

Группа АС-19

Руководитель

Кургасов В.В.

к.п.н.

Липецк 2021

Оглавление

Цель работы	3
Задание кафедры	4
Ход работы	6
Вывод	15
Контрольные вопросы	15

Цель работы

Ознакомиться на практике с понятием процесса в операционной системе.

Приобрести опыт и навыки управления процессами в операционной системе

Linux.

Задание кафедры

Вариант 3.

1. Сгенерировать следующую информацию о m ($m > 2$) процессах системы, имеющих значение идентификатора больше заданного n : флаг — сведения о процессе, статус, PID, PPID, приоритет, использованное время и имя программы.
2. Завершить выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершить с помощью сигнала SIGKILL, задав его имя, второй — с помощью сигнала SIGINT, задав его номер.
3. Через символ « : » вывести идентификаторы процессов, для которых родителем является командный интерпретатор.
4. В отчете предоставьте все шаги ваших действий. То есть следует привести следующее: текст задания, а следом за ним снимок экрана консоли с результатами выполнения задания. Кроме того, перед скриншотом следует привести текстовую запись использованных команд. Кратко поясните результаты выполнения всех команд.
5. Вывести общую информацию о системе
 - 5.1 Вывести информацию о текущем интерпретаторе команд
 - 5.2 Вывести информацию о текущем пользователе
 - 5.3 Вывести информацию о текущем каталоге
 - 5.4 Вывести информацию об оперативной памяти и области подкачки
 - 5.5 Вывести информацию о дисковой памяти
6. Выполнить команды получения информации о процессах

6.1 Получить идентификатор текущего процесса(PID)

6.2 Получить идентификатор родительского процесса(PPID)

6.3 Получить идентификатор процесса инициализации системы

6.4 Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

6.5 Отобразить все процессы

7. Выполнить команды управления процессами

7.1 Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе

7.2 Определить текущее значение nice по умолчанию

7.2 Запустить интерпретатор bash с понижением приоритета

`nice -n 10 bash`

7.3 Определить PID запущенного интерпретатора

7.4 Установить приоритет запущенного интерпретатора равным 5

`renice -n 5 <PID процесса>`

7.5 Получить информацию о процессах bash

`ps lax | grep bash`

Ход работы

Запустим виртуальную машину Linux Ubuntu. Выведем информацию о состоянии процессов системы при помощи команды `ps --sort rss`. Таким образом, список будет выведен в порядке убывания PID

```
author@labserver:~$ ps --sort rss
  PID TTY          TIME CMD
 1148 tty1      00:00:00 ps
   998 tty1      00:00:00 bash
```

Рисунок 1 – Вывод с сортировкой по убыванию

```
540 ?          00:00:00 kmpathd
541 ?          00:00:00 kmpath_handlerd
542 ?          00:00:00 multipathd
551 ?          00:00:00 loop0
553 ?          00:00:00 loop1
554 ?          00:00:00 loop2
556 ?          00:00:00 loop3
558 ?          00:00:00 loop4
560 ?          00:00:00 loop5
564 ?          00:00:00 loop6
565 ?          00:00:00 loop7
579 ?          00:00:00 jbd2/sda2-8
580 ?          00:00:00 ext4-rsv-conver
648 ?          00:00:00 accounts-daemon
651 ?          00:00:00 cron
662 ?          00:00:00 networkd-dispat
666 ?          00:00:00 snapd
667 ?          00:00:00 systemd-logind
672 ?          00:00:00 udisksd
684 tty1       00:00:00 login
714 ?          00:00:00 unattended-upgr
735 ?          00:00:00 polkitd
1112 ?         00:00:00 kworker/u2:1-events_unbound
1139 ?         00:00:00 kworker/0:0-events
675 ?          00:00:00 atd
634 ?          00:00:00 systemd-network
636 ?          00:00:00 systemd-resolve
592 ?          00:00:00 systemd-timesyn
652 ?          00:00:00 dbus-daemon
664 ?          00:00:00 rsyslogd
992 ?          00:00:00 systemd
993 ?          00:00:00 (sd-pam)
998 tty1       00:00:00 bash
1151 tty1       00:00:00 bash
1157 tty1       00:00:00 bash
1165 tty1       00:00:00 ps
author@labserver:~$ _
```

Рисунок 2 – Вывод всех процессов по возрастанию

Завершим выполнение двух процессов, владельцем которых является текущий пользователь. Первый процесс завершаем с помощью сигнала SIGKILL, задав его имя, второй — с помощью сигнала SIGINT, задав его номер.

```
author@labserver:~$ ps
  PID TTY          TIME CMD
 1085 tty1        00:00:00 bash
 1104 tty1        00:00:20 sh
 1107 tty1        00:00:00 vim
 1111 tty1        00:00:00 ps
author@labserver:~$ kill -SIGKILL vim
-bash: kill: vim: arguments must be process or job IDs
author@labserver:~$ pkill -SIGKILL vim
[2]+  Killed                  vim
author@labserver:~$ _
```

Рисунок 3 – Завершение процесса с помощью pkill

```
author@labserver:~$ ps
  PID TTY          TIME CMD
 1085 tty1        00:00:00 bash
 1104 tty1        00:00:43 sh
 1115 tty1        00:00:00 ps
author@labserver:~$ kill -SIGINT 1104
author@labserver:~$ ps
  PID TTY          TIME CMD
 1085 tty1        00:00:00 bash
 1116 tty1        00:00:00 ps
[1]+  Interrupt                sh loop.sh
author@labserver:~$
```

Рисунок 4 – Завершение процесса с помощью сигнала SIGINT

вывести идентификаторы процессов, для которых родителем является командный интерпретатор.

```
author@labserver:~$ ps -o pid --no-headers --sort -comm
1123
1085
```

Рисунок 5 – Вывод идентификаторов процессов, запущенных пользователем

Выведем информацию о текущем интерпретаторе команд с помощью команды `echo $SHELL` и `echo $BASH_VERSION`. Переменная окружения `SHELL` хранит путь до исполняемого файла оболочки.

```
author@labserver:~$ echo $SHELL
/bin/bash
author@labserver:~$ echo $BASH_VERSION
5.0.17(1)-release
author@labserver:~$ _
```

Рисунок 6 – Вывод информации о текущем интерпретаторе команд

Выведем информацию о текущем пользователе. Это можно сделать с помощью команды `whoami`.

```
author@labserver:~$ whoami
author
```

Рисунок 7 – Вывод информации о текущем пользователе

Выведем информацию о текущем каталоге с помощью команд `pwd` и `ls`. `Pwd` показывает путь до текущего каталога, а `ls` выводит его содержимое.

```
author@labserver:~$ pwd
/home/author
author@labserver:~$ ls /home/author
channel loop2.sh loop.sh new res res2
author@labserver:~$
```

Рисунок 8 – Вывод информации о текущем каталоге

Выведем информацию об оперативной памяти и области подкачки с помощью команд `free` и `top`.


```

PID USER      PR  NI   VIRT   RES    SHR  S  %CPU  %MEM    TIME+  COMMAND
 990 author    20   0   2608    596    528  R   99,9   0,0   12:19.86  sh
   1 root       20   0  101928  11408   8452  S    0,0   0,6    0:01.16  systemd
   2 root       20   0        0        0        0  S    0,0   0,0    0:00.00  kthreadd
   3 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  rcu_gp
   4 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  rcu_par_gp
   5 root       20   0        0        0        0  I    0,0   0,0    0:00.06  kworker/0:0-events
   6 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  kworker/0:0H-kblockd
   9 root       0 -20        0        0        0  I    0,0   0,0    0:00.00  mm_percpu_wq
  10 root       20   0        0        0        0  S    0,0   0,0    0:00.06  ksoftirqd/0
  11 root       20   0        0        0        0  I    0,0   0,0    0:00.32  rcu_sched
  12 root       rt    0        0        0        0  S    0,0   0,0    0:00.00  migration/0
  13 root      -51   0        0        0        0  S    0,0   0,0    0:00.00  idle_inject/0
  14 root       20   0        0        0        0  S    0,0   0,0    0:00.00  cpuhp/0
  15 root       20   0        0        0        0  S    0,0   0,0    0:00.00  kdevtmpfs
  16 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  netns
  17 root       20   0        0        0        0  S    0,0   0,0    0:00.00  rcu_tasks_kthre
  18 root       20   0        0        0        0  S    0,0   0,0    0:00.00  kauditd
  19 root       20   0        0        0        0  S    0,0   0,0    0:00.00  khungtaskd
  20 root       20   0        0        0        0  S    0,0   0,0    0:00.00  oom_reaper
  21 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  writeback
  22 root       20   0        0        0        0  S    0,0   0,0    0:00.00  kcompactd0
  23 root       25   5        0        0        0  S    0,0   0,0    0:00.00  ksmd
  24 root       39  19        0        0        0  S    0,0   0,0    0:00.00  khugepaged
  70 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  kintegrityd
  71 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  kblockd
  72 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  blkcg_punt_bio
  73 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  tpm_dev_wq
  74 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  ata_sff
  75 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  md
  76 root        0 -20        0        0        0  I    0,0   0,0    0:00.00  edac-poller
[1]+  Stopped                  top
author@labserver:~$ free
              total        used        free      shared  buff/cache   available
Mem:           2035452       143788       1555468         1036       336196       1736276
Swap:           1779708           0         1779708
author@labserver:~$

```

Рисунок 9 – Вывод информации об оперативной памяти и области подкачки с помощью команды free

```

top - 16:37:31 up 14 min,  1 user,  load average: 1,00, 1,01, 0,74
Tasks:  98 total,   2 running,  95 sleeping,   1 stopped,   0 zombie
%Cpu(s):100,0 us,  0,0 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem : 1987,7 total, 1518,5 free,  140,9 used,  328,4 buff/cache
MiB Swap: 1738,0 total, 1738,0 free,   0,0 used. 1695,1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 209 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kdmflush
 237 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 raid5wq
 284 root       20  0      0      0      0 S   0,0   0,0   0:00.00 jbd2/dm-0-8
 285 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 ext4-rsv-conver
 356 root       19 -1  53348  14264  13256 S   0,0   0,7   0:00.13 systemd-journal
 387 root       20  0  21916   5996  3988 S   0,0   0,3   0:00.53 systemd-udev
 405 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 iprt-VBoxQueue
 537 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kaluad
 538 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kmpath_rdadcd
 539 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kmpathd
 540 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 kmpath_handlerd
 541 root        rt  0 280308  18112   8200 S   0,0   0,9   0:00.05 multipathd
 552 root        0 -20      0      0      0 S   0,0   0,0   0:00.02 loop0
 554 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 loop1
 555 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 loop2
 560 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 loop3
 561 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 loop4
 562 root        0 -20      0      0      0 S   0,0   0,0   0:00.04 loop5
 565 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 loop6
 569 root        0 -20      0      0      0 S   0,0   0,0   0:00.00 loop7
 578 root       20  0      0      0      0 S   0,0   0,0   0:00.00 jbd2/sda2-8
 579 root        0 -20      0      0      0 I   0,0   0,0   0:00.00 ext4-rsv-conver
 592 systemd+   20  0   90228   6108   5344 S   0,0   0,3   0:00.03 systemd-timesyn
 631 systemd+   20  0   26604   7516   6648 S   0,0   0,4   0:00.03 systemd-network
 633 systemd+   20  0   23896  12112   8192 S   0,0   0,6   0:00.05 systemd-resolve
 645 root       20  0  239296   9316   8356 S   0,0   0,5   0:00.04 accounts-daemon
 648 root       20  0    6812   2788   2580 S   0,0   0,1   0:00.00 cron
 649 message+   20  0    7512   4656   3920 S   0,0   0,2   0:00.04 dbus-daemon
 658 root       20  0   29012  18068  10412 S   0,0   0,9   0:00.07 networkd-dispat
 659 syslog     20  0  224500   5000   3940 S   0,0   0,2   0:00.01 rsyslogd

```

Рисунок 10 – Вывод информации об оперативной памяти и области подкачки с помощью команды top

Выведем информацию о дисковой памяти с помощью команд `ls -l /dev/` и `df`.

```

crw-rw---- 1 root tty      7, 131 ноя 22 16:22 vcsa3
crw-rw---- 1 root tty      7, 132 ноя 22 16:22 vcsa4
crw-rw---- 1 root tty      7, 133 ноя 22 16:22 vcsa5
crw-rw---- 1 root tty      7, 134 ноя 22 16:22 vcsa6
crw-rw---- 1 root tty      7, 64 ноя 22 16:22 vcsu
crw-rw---- 1 root tty      7, 65 ноя 22 16:22 vcsu1
crw-rw---- 1 root tty      7, 66 ноя 22 16:22 vcsu2
crw-rw---- 1 root tty      7, 67 ноя 22 16:22 vcsu3
crw-rw---- 1 root tty      7, 68 ноя 22 16:22 vcsu4
crw-rw---- 1 root tty      7, 69 ноя 22 16:22 vcsu5
crw-rw---- 1 root tty      7, 70 ноя 22 16:22 vcsu6
drwxr-xr-x 2 root root     60 ноя 22 16:22 vfio
crw----- 1 root root    10, 63 ноя 22 16:22 vga_arbiter
crw----- 1 root root    10, 137 ноя 22 16:22 vhci
crw----- 1 root root    10, 238 ноя 22 16:22 vhost-net
crw----- 1 root root    10, 241 ноя 22 16:22 vhost-vsock
crw-rw-rw- 1 root root      1, 5 ноя 22 16:22 zero
crw----- 1 root root    10, 249 ноя 22 16:22 zfs
author@labserver:~$ df
Filesystem                1K-blocks    Used Available Use% Mounted on
udev                      972572         0   972572    0% /dev
tmpfs                     203548      1036   202512    1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 9219412 4497720  4233656   52% /
tmpfs                    1017724         0   1017724    0% /dev/shm
tmpfs                     5120          0     5120    0% /run/lock
tmpfs                    1017724         0   1017724    0% /sys/fs/cgroup
/dev/loop0                63360      63360         0 100% /snap/core20/1242
/dev/loop2                63360      63360         0 100% /snap/core20/1169
/dev/loop1                56832      56832         0 100% /snap/core18/2253
/dev/loop4                68864      68864         0 100% /snap/lxd/21835
/dev/loop3                68864      68864         0 100% /snap/lxd/21545
/dev/loop5                33280      33280         0 100% /snap/snapd/13640
/dev/loop6                33280      33280         0 100% /snap/snapd/13270
/dev/loop7                56832      56832         0 100% /snap/core18/2128
/dev/sda2                 999320    108572   821936   12% /boot
tmpfs                     203544         0   203544    0% /run/user/1000
author@labserver:~$

```

Рисунок 11 – Вывод информации о дисковой памяти с помощью команд `ls -l /dev/` и `df`.

Получим идентификатор текущего процесса(PID). Самый распространённый способ узнать PID Linux - использовать команду `ps`, которая выведет все текущие процессы и их PID.

```

author@labserver:~$ ps
  PID TTY          TIME CMD
 1085 tty1        00:00:00 bash
  1152 tty1        00:00:00 top
  1154 tty1        00:00:00 top
  1175 tty1        00:00:00 ps
author@labserver:~$

```

Рисунок 12 – Получение PID процесса

Получим идентификатор родительского процесса(PPID) с помощью `ps -f`.

```
author@labserver:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
author       1085      996  0  16:25 tty1        00:00:00 -bash
author       1152     1085  0  16:36 tty1        00:00:00 top
author       1154     1085  0  16:37 tty1        00:00:00 top
author       1177     1085  0  16:41 tty1        00:00:00 ps -f
author@labserver:~$
```

Рисунок 13 – Получение PPID

Получим идентификатор процесса инициализации системы. Поскольку этим процессом является `init`, то его PID можно легко узнать с помощью команды `pidof имя_процесса`.

```
author@labserver:~$ pidof init
1
author@labserver:~$ _
```

Рисунок 14 – Получение PID процесса инициализации системы

Получим информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд. Сделать это можно с помощью команды `ps -a -f`.

```
author@labserver:~$ ps -a -f
UID          PID     PPID  C  STIME TTY          TIME CMD
author       1085      996  0  16:25 tty1        00:00:00 -bash
author       1152     1085  0  16:36 tty1        00:00:00 top
author       1154     1085  0  16:37 tty1        00:00:00 top
author       1249     1085  0  17:23 tty1        00:00:00 ps -a -f
author@labserver:~$
```

Рисунок 15 - Получение информации о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд.

Чтобы отобразить все процессы используем `ps -e`.

```

552 ?      00:00:00 loop0
554 ?      00:00:00 loop1
555 ?      00:00:00 loop2
560 ?      00:00:00 loop3
561 ?      00:00:00 loop4
562 ?      00:00:00 loop5
565 ?      00:00:00 loop6
569 ?      00:00:00 loop7
578 ?      00:00:00 jbd2/sda2-8
579 ?      00:00:00 ext4-rsv-conver
592 ?      00:00:00 systemd-timesyn
631 ?      00:00:00 systemd-network
633 ?      00:00:00 systemd-resolve
645 ?      00:00:00 accounts-daemon
648 ?      00:00:00 cron
649 ?      00:00:00 dbus-daemon
658 ?      00:00:00 networkd-dispat
659 ?      00:00:00 rsyslogd
662 ?      00:00:00 snapd
663 ?      00:00:00 systemd-logind
668 ?      00:00:00 udisksd
669 ?      00:00:00 atd
670 ?      00:00:01 kworker/0:4-events
703 ?      00:00:00 unattended-upgr
719 ?      00:00:00 polkitd
969 ?      00:00:00 systemd
971 ?      00:00:00 (sd-pam)
990 ?      00:59:04 sh
996 tty1    00:00:00 login
1085 tty1    00:00:00 bash
1152 tty1    00:00:00 top
1154 tty1    00:00:00 top
1225 ?      00:00:00 kworker/0:0-cgroup_destroy
1231 ?      00:00:00 kworker/u2:2-events_power_efficient
1235 ?      00:00:00 kworker/u2:0-events_unbound
1250 tty1    00:00:00 ps
author@labserver:~$

```

Рисунок 16 – Отображение всех процессов.

Определим текущее значение nice по умолчанию. По умолчанию nice устанавливает значение приоритета 10.

```

author@labserver:~$ ps -l
F S  UID      PID      PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000     1085        996  0  80   0 - 2099 do_wai tty1        00:00:00 bash
0 T   1000     1152     1085  0  80   0 - 2285 do_sig tty1        00:00:00 top
0 T   1000     1154     1085  0  80   0 - 2285 do_sig tty1        00:00:00 top
0 R   1000     1251     1085  0  80   0 - 2203 -      tty1        00:00:00 ps
author@labserver:~$

```

Рисунок 17 – Определение nice

Запустим интерпретатор bash с понижением приоритета используя nice -n 10 bash. Для этого загрузимся как пользователь с root правами и посмотрим имеющиеся процессы.

```
author@labserver:~$ ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000     1085      996  0  80   0 - 2099 do_wai tty1        00:00:00 bash
0 T   1000     1152     1085  0  80   0 - 2285 do_sig tty1        00:00:00 top
0 T   1000     1154     1085  0  80   0 - 2285 do_sig tty1        00:00:00 top
0 R   1000     1251     1085  0  80   0 - 2203 -      tty1        00:00:00 ps
author@labserver:~$ nice -n 10 bash
```

Рисунок 18 – Запуск bash с понижением приоритета

Определим PID запущенного интерпретатора.

```
author@labserver:~$ ps -l
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000     1085      996  0  80   0 - 2099 do_wai tty1        00:00:00 bash
0 T   1000     1152     1085  0  80   0 - 2285 do_sig tty1        00:00:00 top
0 T   1000     1154     1085  0  80   0 - 2285 do_sig tty1        00:00:00 top
0 S   1000     1252     1085  0  90  10 - 2068 do_wai tty1        00:00:00 bash
0 R   1000     1259     1252  0  90  10 - 2203 -      tty1        00:00:00 ps
author@labserver:~$ _
```

Рисунок 19 – Определение PID bash

Установим приоритет запущенного интерпретатора равным 5 с помощью renice -n 5 <PID процесса>.

```
author@labserver:~$ renice -n 5 1085
1085 (process ID) old priority 0, new priority 5
author@labserver:~$ _
```

Рисунок 20 – Установка нового приоритета

Получение информации о процессах bash с помощью ps lax | grep bash.

```
author@labserver:~$ ps lax | grep bash
4 1000 1085 996 25 5 8396 5300 do_wai SN tty1 0:00 -bash
0 1000 1252 1085 30 10 8272 5084 do_wai SN tty1 0:00 bash
0 1000 1264 1252 30 10 6300 728 pipe_w SN+ tty1 0:00 grep --color=auto bash
author@labserver:~$
```

Рисунок 21 – Получение информации о процессах

Вывод

В ходе выполнения лабораторной работы я 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 Ubuntu.

Задача переходит в состояние running (выполнения) после выделения ей процессора. При блокировке задача переходит в состояние sleeping (спячки), а при остановке работы в состояние останов (stopped). Состояние zombie (зомби) показывает, что выполнение задачи прекратилось, однако она еще не была удалена из системы. Например, если процесс состоит из нескольких потоков, он будет пребывать в состоянии зомби, пока все потоки не получат уведомление о завершении работы основного процесса. Задача в состоянии dead (смерти) может быть удалена из системы. Состояния active (активный) и expired (неактивный) используются при планировании выполнения процесса, и поэтому они не сохраняются в переменной state.

2. Как создаются задачи в ОС Ubuntu?

Задачи создаются путем вызова функции clone.

3. Назовите классы потоков ОС Ubuntu

1.Потоки реального времени, обслуживаемые по алгоритму FIFO.

2.Потоки реального времени, обслуживаемые в порядке циклической очереди.

3.Потоки разделения времени

4. Как используется приоритет планирования при запуске задачи

Для Linux приоритет планирования процесса реального времени варьируется от 1 (самый низкий приоритет) до 99 (самый высокий приоритет). Обычные процессы также имеют приоритет планирования, равный 0.

5. Как можно изменить приоритет для выполняющейся задачи?

С помощью `renice`

`renice` — UNIX-утилита, позволяющая изменить приоритет запущенных задач. Привилегированный пользователь (`root`) может указать отрицательное смещение. Команда `renice` может смещать приоритет в диапазоне от -20 (наивысший приоритет) до 19 (низший приоритет) от текущего. Для изменения значения приоритета отдельных процессов достаточно перечислить их идентификаторы.