ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»
В Г. СМОЛЕНСКЕ

Кафедра вычислительной техники

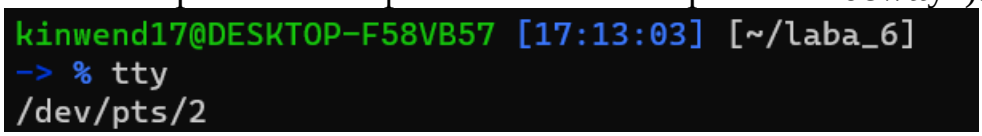
Дисциплина: ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
АВТОМАТИЗИРОВАННЫХ СИСТЕМ

Лабораторная работа № 6.
ПЛАНИРОВАНИЕ ЗАДАНИЙ И УПРАВЛЕНИЕ ПРИОРИТЕТАМИ ПРОЦЕССОВ

Студент: Старостенков А.А.
Группа: ВМ-22 (маг.)
Вариант: 15
Преподаватель: Федулов Я.А.

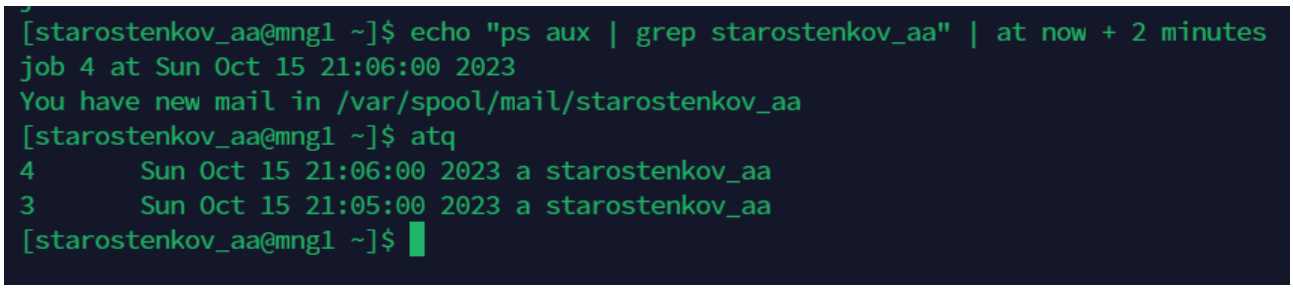
СМОЛЕНСК
2023

1. Определите имя файла текущего терминала командой tty (при работе в текстовом режиме имя файла основного терминала /dev/tty1).



2. Создайте отложенное на 2 минуты задание, выводящее подробный список всех процессов пользователя student (в моём случае kinwend17) на экран.
3. Дождитесь вывода списка процессов на экран.

Сразу стоит оговориться: отложенная команда выполняется, но в терминале локальной машины не выводится сообщение о выполнении. Команда «at» использует почтовый сервер для вывода уведомления о выполнении. Для примера, сделаем выполнение команды на вычислительном кластере СФ МЭИ. Наше письмо (уведомление) будет храниться в файле /var/spool/mail/starostenkov_aa. Для локальной машины будет дополнять команды перенаправлением вывода в файл, чтобы смотреть результаты работы.



```
[starostenkov_aa@mng1 ~]$ cat /var/spool/mail/starostenkov_aa
From starostenkov_aa@mng1.sbmpei  Sun Oct 15 17:44:00 2023
Return-Path: <starostenkov_aa@mng1.sbmpei>
X-Original-To: starostenkov_aa
Delivered-To: starostenkov_aa@mng1.sbmpei
Received: by mng1.sbmpei (Postfix, from userid 1379)
        id 607A8C00C0; Sun, 15 Oct 2023 17:44:00 +0300 (MSK)
Subject: Output from your job        1
To: starostenkov_aa@mng1.sbmpei
Message-Id: <20231015144400.607A8C00C0@mng1.sbmpei>
Date: Sun, 15 Oct 2023 17:44:00 +0300 (MSK)
From: starostenkov_aa@mng1.sbmpei

/home/students/vm-19m/starostenkov_aa

From starostenkov_aa@mng1.sbmpei  Sun Oct 15 21:04:00 2023
Return-Path: <starostenkov_aa@mng1.sbmpei>
X-Original-To: starostenkov_aa
Delivered-To: starostenkov_aa@mng1.sbmpei
Received: by mng1.sbmpei (Postfix, from userid 1379)
        id 63E58C00C1; Sun, 15 Oct 2023 21:04:00 +0300 (MSK)
Subject: Output from your job        2
To: starostenkov_aa@mng1.sbmpei
Message-Id: <20231015180400.63E58C00C1@mng1.sbmpei>
Date: Sun, 15 Oct 2023 21:04:00 +0300 (MSK)
From: starostenkov_aa@mng1.sbmpei

starost+ 11843  0.0  0.0 112716   952 ?        SN   21:03   0:00 grep kinwend17

From starostenkov_aa@mng1.sbmpei  Sun Oct 15 21:05:00 2023
Return-Path: <starostenkov_aa@mng1.sbmpei>
X-Original-To: starostenkov_aa
Delivered-To: starostenkov_aa@mng1.sbmpei
Received: by mng1.sbmpei (Postfix, from userid 1379)
        id B06ECC00C2; Sun, 15 Oct 2023 21:05:00 +0300 (MSK)
Subject: Output from your job        3
To: starostenkov_aa@mng1.sbmpei
Message-Id: <20231015180500.B06ECC00C2@mng1.sbmpei>
Date: Sun, 15 Oct 2023 21:05:00 +0300 (MSK)
From: starostenkov_aa@mng1.sbmpei

starost+ 11962  0.0  0.0 112716   948 ?        SN   21:05   0:00 grep kinwend17

From starostenkov_aa@mng1.sbmpei  Sun Oct 15 21:06:00 2023
Return-Path: <starostenkov_aa@mng1.sbmpei>
From starostenkov_aa@mng1.sbmpei  Sun Oct 15 21:06:00 2023
Return-Path: <starostenkov_aa@mng1.sbmpei>
X-Original-To: starostenkov_aa
Delivered-To: starostenkov_aa@mng1.sbmpei
Received: by mng1.sbmpei (Postfix, from userid 1379)
        id D7E72C00C1; Sun, 15 Oct 2023 21:06:00 +0300 (MSK)
Subject: Output from your job        4
To: starostenkov_aa@mng1.sbmpei
Message-Id: <20231015180600.D7E72C00C1@mng1.sbmpei>
Date: Sun, 15 Oct 2023 21:06:00 +0300 (MSK)
From: starostenkov_aa@mng1.sbmpei

root     11033  0.0  0.0 121812  5336 ?        Ss   20:57   0:00 sshd: starostenkov_aa [priv]
starost+ 11038  0.0  0.0 121952  2228 ?        S    20:57   0:00 sshd: starostenkov_aa@pts/1
starost+ 12078  0.0  0.0 112716   952 ?        SN   21:06   0:00 grep starostenkov_aa

[starostenkov_aa@mng1 ~]$
```

```
kinwend17@DESKTOP-F58VB57 [21:20:50] [~/laba_6]
-> % ll
total 4.0K
-rw-r--r-- 1 kinwend17 kinwend17  0 Oct 15 21:19 lb6_script.txt
-rw-r--r-- 1 kinwend17 kinwend17 82 Oct 15 21:22 task2.txt
kinwend17@DESKTOP-F58VB57 [21:22:37] [~/laba_6]
-> % cat task2.txt
kinwend+     768  0.0  0.0   8168   648 ?        SN   21:22   0:00 grep kinwend17
kinwend17@DESKTOP-F58VB57 [21:22:44] [~/laba_6]
-> %
```

4. Установите отложенные задания, выводящие сообщения: «конец пары», «обед», «пора спать!!!» (с одновременным выключением компьютера командой halt — только для суперпользователя root).

```
kinwend17@DESKTOP-F58VB57 [21:29:39] [~/laba_6]
-> % echo "'конец пары' > task3_1.txt" | at now + 5 minutes
warning: commands will be executed using /bin/sh
job 21 at Sun Oct 15 21:34:00 2023
kinwend17@DESKTOP-F58VB57 [21:29:53] [~/laba_6]
-> % echo "'обед' > task3_2.txt" | at now + 2 hours
warning: commands will be executed using /bin/sh
job 22 at Sun Oct 15 23:30:00 2023
kinwend17@DESKTOP-F58VB57 [21:30:01] [~/laba_6]
-> %
kinwend17@DESKTOP-F58VB57 [21:30:31] [~/laba_6]
-> % echo "'пора спать!!!' > task3_3.txt" | at now + 8 hours
kinwend17@DESKTOP-F58VB57 [21:31:04] [~/laba_6]
-> % echo "'пора спатьecho "'обед' > task3_2.txt" | at now + 2
warning: commands will be executed using /bin/sh
job 23 at Mon Oct 16 05:31:00 2023
kinwend17@DESKTOP-F58VB57 [21:31:16] [~/laba_6]
-> % atq
22      Sun Oct 15 23:30:00 2023 a kinwend17
21      Sun Oct 15 21:34:00 2023 a kinwend17
23      Mon Oct 16 05:31:00 2023 a kinwend17
kinwend17@DESKTOP-F58VB57 [21:31:26] [~/laba_6]
-> % echo "sudo halt > task3_4.txt" | at now + 10 hours
warning: commands will be executed using /bin/sh
job 24 at Mon Oct 16 07:31:00 2023
kinwend17@DESKTOP-F58VB57 [21:31:36] [~/laba_6]
-> %
```

5. Удалите задание, выводящее на экран сообщение «обед».

```
kinwend17@DESKTOP-F58VB57 [21:31:36] [~/laba_6]
-> % atrm 22
kinwend17@DESKTOP-F58VB57 [21:32:35] [~/laba_6]
-> % atq
24      Mon Oct 16 07:31:00 2023 a kinwend17
21      Sun Oct 15 21:34:00 2023 a kinwend17
23      Mon Oct 16 05:31:00 2023 a kinwend17
kinwend17@DESKTOP-F58VB57 [21:32:38] [~/laba_6]
-> %
```

6. Создайте задание, выводящее на экран текущее время в формате ЧЧ:ММ каждую минуту.

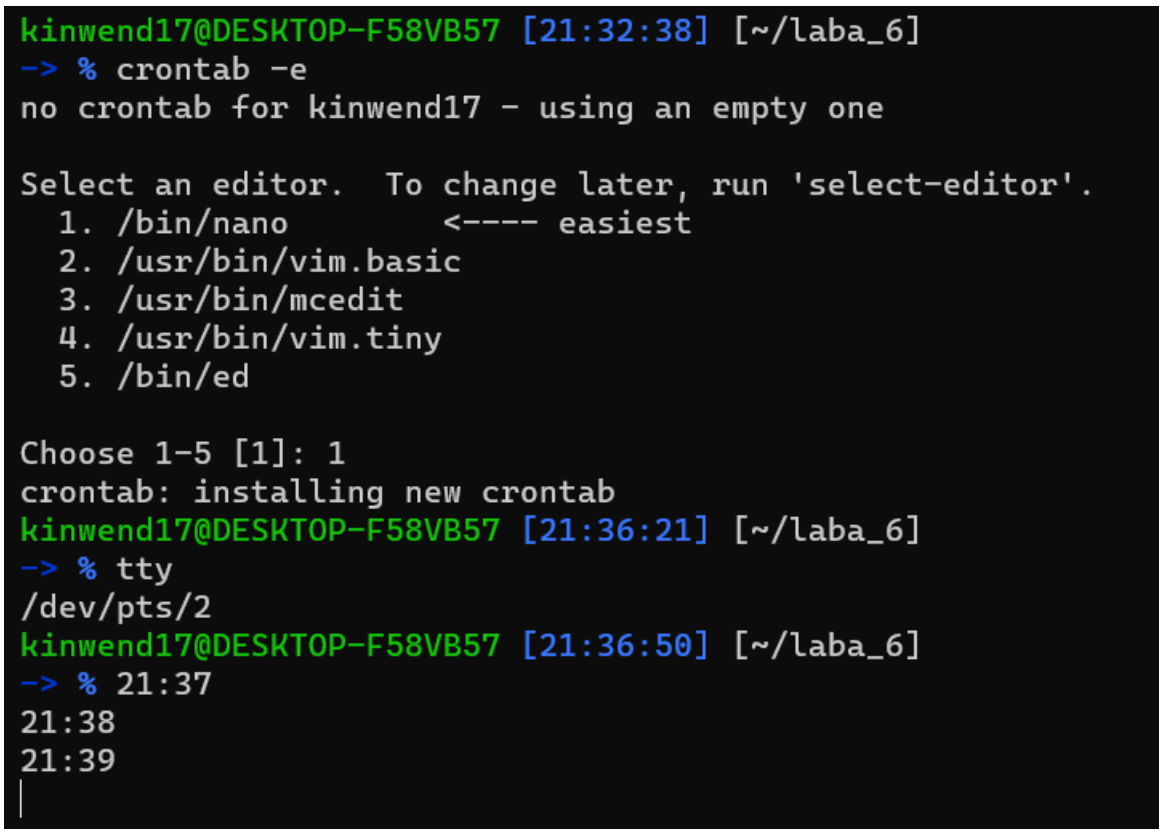
```
kinwend17@DESKTOP-F58VB57 [21:32:38] [~/laba_6]
-> % crontab -e
no crontab for kinwend17 - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/mcedit
  4. /usr/bin/vim.tiny
  5. /bin/ed

Choose 1-5 [1]: 1
crontab: installing new crontab
kinwend17@DESKTOP-F58VB57 [21:36:21] [~/laba_6]
-> %
```

```
  GNU nano 4.8                                  /tmp/crontab.f0eioe/crontab
* * * * * date +\%H:\%M > /dev/pts/2

# 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
```



7. Напишите и скомпилируйте программу output, которая в бесконечном цикле выводит символ «a» в файл, имя которого указано в первом аргументе командной строки.

Программу напишем на языке C, код будем писать с помощью VS Code, скомпилируем с помощью gcc.

Gcc outpuc.c -o output

Программа вывода "a" в файл бесконечно

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <output_file_name>\n", argv[0]);
        return 1;
    }

    const char *outputFileName = argv[1];
    FILE *outputFile = fopen(outputFileName, "w");

    if (outputFile == NULL) {
        perror("Error opening the output file");
        return 1;
    }

    int character = 'a';

    while (1) {
        fputc(character, outputFile);
        fflush(outputFile);  // Принудительно сбрасываем буфер вывода
    }

    fclose(outputFile);
    return 0;
}

```



```
kinwend17@DESKTOP-F58VB57 [21:59:24] [~/laba_6]
-> % rm -rf test.txt
kinwend17@DESKTOP-F58VB57 [21:59:27] [~/laba_6]
-> % ./output test.txt
^C
kinwend17@DESKTOP-F58VB57 [21:59:31] [~/laba_6]
-> % cat test.txt
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

8. Запустите одновременно два фоновых процесса и через некоторое время принудительно завершите их.
Запуск:
./output a.txt & ./output b.txt &
Завершение:
kill PID1 & kill PID2 &
где PID1 и PID2 — идентификаторы двух запущенных процессов.

```
kinwend17@DESKTOP-F58VB57 [22:02:16] [~/laba_6]
-> % ./output a.txt & ./output b.txt &
[1] 7439
[2] 7440
kinwend17@DESKTOP-F58VB57 [22:02:35] [~/laba_6]
-> % jobs
[1]  - running    ./output a.txt
[2]  + running    ./output b.txt
kinwend17@DESKTOP-F58VB57 [22:02:47] [~/laba_6]
-> % ll
total 45M
-rw-r--r-- 1 kinwend17 kinwend17  16M Oct 15 22:02  a.txt
-rw-r--r-- 1 kinwend17 kinwend17  16M Oct 15 22:02  b.txt
```

9. Сравните размер файлов a.txt и b.txt.

```
kinwend17@DESKTOP-F58VB57 [22:02:50] [~/laba_6]
-> % kill 7439 && kill 7440
[1]  - 7439 terminated  ./output a.txt
[2]  + 7440 terminated  ./output b.txt
kinwend17@DESKTOP-F58VB57 [22:03:39] [~/laba_6]
-> % ls -al
total 145368
drwxr-xr-x  2 kinwend17 kinwend17     4096 Oct 15 22:02  .
drwxr-xr-x 12 kinwend17 kinwend17     4096 Oct 15 22:03  ..
-rw-r--r--  1 kinwend17 kinwend17 67774290 Oct 15 22:03  a.txt
-rw-r--r--  1 kinwend17 kinwend17 68032278 Oct 15 22:03  b.txt
```

Файл a.txt имеет размер 67 774 290 байт, а файл b.txt имеет размер 68 032 278 байт. Файл b.txt больше по размеру на 257 988 байт.

Размеры файлов a.txt и b.txt отличаются, потому что мы выполнили два фоновых процесса, каждый из которых запускал программу output, которая бесконечно добавляла символ «a» в соответствующий файл. В процессе выполнения, один из процессов мог завершиться быстрее другого, что привело к разнице в размере файлов. Это нормальное поведение, так как бесконечные процессы завершаются непредсказуемо, и файлы могут закрыться с разными данными.

Итак, файл b.txt больше по размеру, потому что один из фоновых процессов, запущенных для output b.txt, успел записать больше символов «a» в файл, чем процесс для output a.txt.

10. Запустите одновременно два фоновых процесса с разными приоритетами (0 и +10 соответственно) и через некоторое время принудительно завершите их.

```
kinwend17@DESKTOP-F58VB57 [22:10:16] [~/laba_6]
-> % nice -n 0 ./output a.txt & nice -n 10 ./output b.txt &
[1] 7470
[2] 7471
kinwend17@DESKTOP-F58VB57 [22:10:21] [~/laba_6]
-> % pkill -f "output a.txt" && pkill -f "output b.txt"
[1]  - 7470 terminated  nice -n 0 ./output a.txt
[2]  + 7471 terminated  nice -n 10 ./output b.txt
kinwend17@DESKTOP-F58VB57 [22:11:49] [~/laba_6]
-> % jobs
kinwend17@DESKTOP-F58VB57 [22:11:54] [~/laba_6]
-> % ls -al
total 195364
drwxr-xr-x  2 kinwend17 kinwend17     4096 Oct 15 22:02  .
drwxr-xr-x 12 kinwend17 kinwend17     4096 Oct 15 22:11  ..
-rw-r--r--  1 kinwend17 kinwend17 93257181 Oct 15 22:11  a.txt
-rw-r--r--  1 kinwend17 kinwend17 93741692 Oct 15 22:11  b.txt
```

Размер файлов a.txt и b.txt, как видно из ваших данных, следующий:

- a.txt: 93,257,181 байт.
- b.txt: 93,741,692 байта.

Размер b.txt больше, чем a.txt.

Скорее всего, процесс с более высоким приоритетом (nice -n 10) успел записать больше символов «a» в файл b.txt, чем процесс с нулевым приоритетом (nice -n 0) в файл a.txt. Это объясняет разницу в размерах файлов a.txt и b.txt.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1) Что осуществляет планировщик cron?

Планировщик cron в Linux осуществляет периодическое выполнение задач (команд, скриптов) в определенное время или с определенной периодичностью. Это позволяет автоматизировать выполнение регулярных заданий, таких как резервное копирование, обновление данных и многие другие операции.

2) Какие основные команды для работы с планировщиком заданий в Linux

Основные команды для работы с планировщиком заданий в Linux:

- crontab: Команда для управления cron-заданиями пользователя. Можно создавать, редактировать и удалять задания.
- cron: Демон-планировщик, который управляет выполнением cron-заданий.

- at: Команда для выполнения отложенных заданий один раз в указанный момент времени.
- nice: Команда для изменения приоритета выполнения процессов.

3) Что происходит с выводимой отложенным процессом информацией?

Выводимая информация отложенным процессом в случае использования at отправляется по электронной почте на почтовый ящик пользователя.

4) Для чего нужна команда at? Какой у ней формат?

Команда at в Linux используется для планирования выполнения заданий в определенный момент времени или через определенное время. Её формат обычно такой: **at [опции] время**. Опции позволяют уточнить, когда и как выполнять задание. Затем вводится команда, которую нужно выполнить.

5) С помощью каких команд осуществляется управление приоритетами процессов в Linux?

Для управления приоритетами процессов в Linux используются следующие команды:

- nice: устанавливает приоритет выполнения процесса.
- renice: изменяет приоритет существующего процесса.

6) Каково влияние приоритета на производительность процесса?

Приоритет процесса влияет на то, насколько «жадно» процесс использует процессорное время. Процессам с более высоким приоритетом предоставляется больше процессорного времени, и они выполняются быстрее, в то время как процессы с низким приоритетом могут выполняться медленнее. Установка правильного приоритета позволяет более эффективно управлять ресурсами процессора в системе.