

Лабораторная работа №12

ДИОН ГОНССАН СЕДРИК МИШЕЛ

¹RUDN University, Moscow, Russian Federation

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.

2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

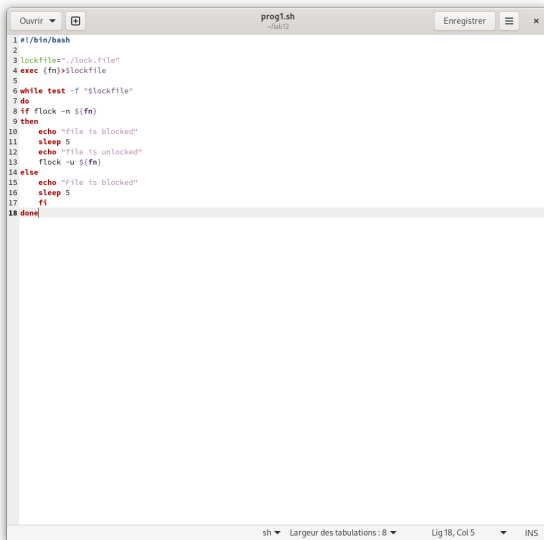
Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна (**Prog:bash?**).

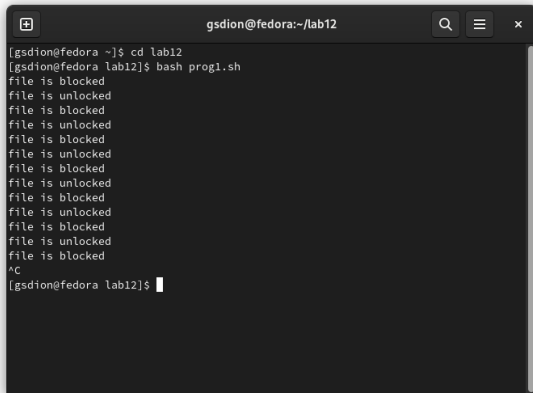
1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов. (рис. (fig:001?; fig:002?))

Выполнение лабораторной работы



```
1 #!/bin/bash
2
3 lockfile="./lock.file"
4 exec {fn}>$lockfile
5
6 while test -f "$lockfile"
7 do
8   if flock -n ${fn}
9   then
10     echo "file is blocked"
11     sleep 5
12     echo "file is unlocked"
13     flock -u ${fn}
14   else
15     echo "file is blocked"
16     sleep 5
17   fi
18 done
```

Рис. 1: Текст первой программы

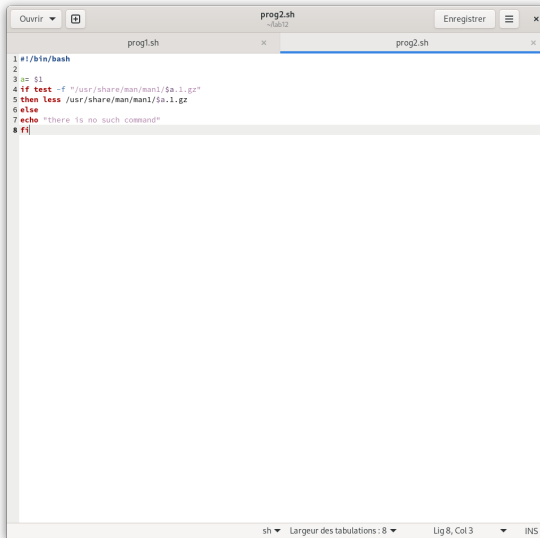


```
gsdion@fedora: ~/lab12
[gsdion@fedora ~]$ cd lab12
[gsdion@fedora lab12]$ bash prog1.sh
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
^C
[gsdion@fedora lab12]$
```

Рис. 2: Результат

2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`. (рис. (fig:003?; fig:004?; fig:005?))

Выполнение лабораторной работы



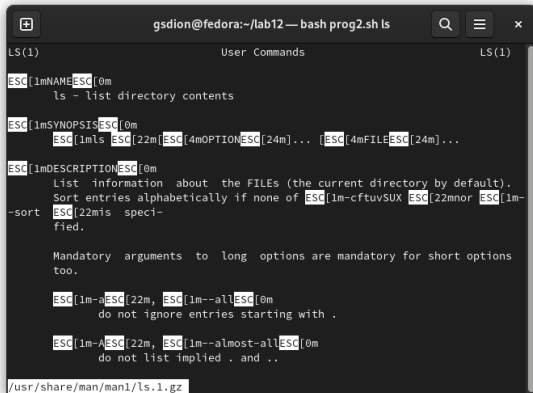
The image shows a screenshot of a text editor window titled "prog2.sh" with a subtitle "~ / tab12". The window has a menu bar with "Ouvrir", "Enregistrer", and a close button. Below the menu bar, there are two tabs: "prog1.sh" and "prog2.sh", with "prog2.sh" being the active tab. The main area of the window contains the following text:

```
1 #!/bin/bash
2
3 a= $1
4 if test -f "/usr/share/man/man1/$a.1.gz"
5 then less /usr/share/man/man1/$a.1.gz
6 else
7 echo "there is no such command"
8 fi
```

At the bottom of the window, there is a status bar that reads "sh", "Largeur des tabulations : 8", "Lig 8, Col 3", and "INS".

Рис. 3: Текст второй программы

Выполнение лабораторной работы



```
gsdion@fedora:~/lab12 — bash prog2.sh ls
LS(1)                                User Commands                                LS(1)

ESC[1mNAMEESC[0m
    ls - list directory contents

ESC[1mSYNOPSISESC[0m
    ESC[1mls ESC[22mESC[4mOPTIONESC[24m]... ESC[4mFILEESC[24m]...

ESC[1mDESCRIPTIONESC[0m
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of ESC[1m-cftuvSUX ESC[22mnor ESC[1m-
-sort ESC[22mis speci-
    fied.

    Mandatory arguments to long options are mandatory for short options
    too.

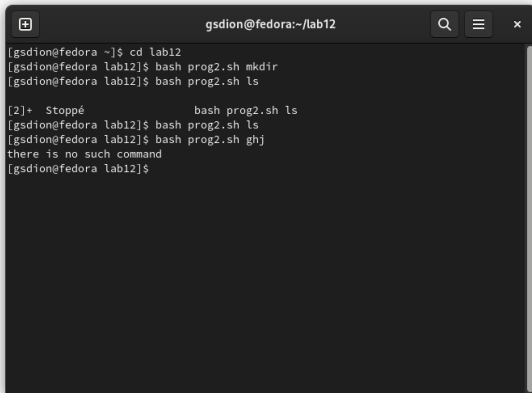
    ESC[1m-aESC[22m, ESC[1m--allESC[0m
        do not ignore entries starting with .

    ESC[1m-AESC[22m, ESC[1m--almost-allESC[0m
        do not list implied . and ..

/usr/share/man/man1/ls.1.gz
```

Рис. 4: Результат

Выполнение лабораторной работы

A terminal window with a dark background and light text. The title bar at the top reads "gsdion@fedora:~/lab12". The terminal shows a sequence of commands and their outputs. The user enters "cd lab12", then "bash prog2.sh mkdir", and "bash prog2.sh ls". Then, they press Ctrl+C, which results in "[2]+ Stoppé" and "bash prog2.sh ls". Finally, they enter "bash prog2.sh ls", "bash prog2.sh ghj", and "ls", with the last command resulting in the error "there is no such command".

```
gsdion@fedora ~]$ cd lab12
gsdion@fedora lab12]$ bash prog2.sh mkdir
gsdion@fedora lab12]$ bash prog2.sh ls

[2]+  Stoppé                               bash prog2.sh ls
gsdion@fedora lab12]$ bash prog2.sh ls
gsdion@fedora lab12]$ bash prog2.sh ghj
there is no such command
gsdion@fedora lab12]$
```

Рис. 5: Результат


```
[gsdion@fedora lab12]$ bash prog3.sh 15
kxrhkxmzhqptd
[gsdion@fedora lab12]$ bash prog3.sh 30
jgabjwumdxscsqbfjrkcahbfamt
[gsdion@fedora lab12]$ bash prog3.sh 65
fahoykgohbtvritykloyfpqneusadjucitrneifsjnokrvrznqkyklpgdobawug1
[gsdion@fedora lab12]$ clea
```

Рис. 7: Результат

В процессе выполнения этой лабораторной работы я продолжила осваивать программирование на `bash`.

Спасибо за внимание!