

Лабораторная работа 12

Программирование в командном процессоре ОС UNIX.
Расширенное программирование

Aristova A.O.

2022, 28 May 2022

RUDN 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, 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`.

- 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.

Преимущества и недостатки Bash:

Многие языки программирования намного удобнее и понятнее для пользователя. Например, Python более быстр, так как компилируется байтами. Однако главное преимущество Bash – его повсеместное распространение. Более того, Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от других языков программирования). Но относительно таких `bash` очень сжат. То есть, например, C имеет гораздо более широкие возможности для разработчика.

Выполнение лабораторной работы

1. Я написала командный файл, реализующий упрощенный механизм семафоров.

```
lockfile="./locking.file"
exec {fn}>"$lockfile"
if test -f "$lockfile"
then
    while [ 1 != 0 ]
    do
        if flock -n ${fn}
        then
            echo "file was locked"
            sleep 4
            echo "unlocking"
            flock -u ${fn}

        else
            echo "file was unlocked"
            sleep 3
        fi
    done
fi
```

Рис. 1: Скрипт к заданию 1.

Затем я добавила право на исполнение файла и выполнила его.

```
[aoaristova@fedora lab12]$ vi lab12_1  
[aoaristova@fedora lab12]$ chmod +x lab12_1  
[aoaristova@fedora lab12]$ ./lab12_1  
flock: requires file descriptor, file or directory  
file was unlocked
```

Рис. 2: Результат выполнения скрипта 1.

2. Я просмотрела содержимое каталога
/usr/share/man/man1.

```
[aoaristova@fedora lab12]$ cd /usr/share/man/man1
[aoaristova@fedora man1]$ ls
.:1.gz
'[:1.gz'
ab.1.gz
abrt.1.gz
abrt-action-analyze-backtrace.1.gz
abrt-action-analyze-c.1.gz
abrt-action-analyze-ccpp-local.1.gz
abrt-action-analyze-core.1.gz
abrt-action-analyze-java.1.gz
abrt-action-analyze-oops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-vmcore.1.gz
abrt-action-analyze-vulnerability.1.gz
abrt-action-analyze-xorg.1.gz
abrt-action-check-oops-for-hw-error.1.gz
abrt-action-find-bodhi-update.1.gz
abrt-action-generate-backtrace.1.gz
abrt-action-generate-core-backtrace.1.gz
```

Рис. 3: Просмотр каталога /usr/share/man/man1.

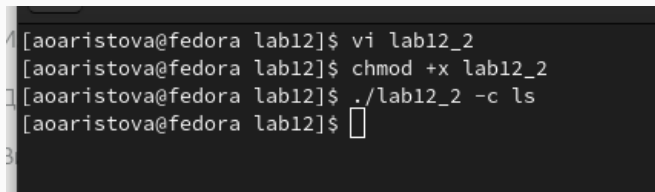
Выполнение лабораторной работы

Я написала командный файл, позволяющий реализовать команду `man` с помощью команды `less`, которая выдает содержимое справки по команде.

```
И command=""
Д while getopts :c: opt
Д do
В case $opt in
  [c]) command="$OPTARG";;
Д esac
Д done
З
И if test -f "/usr/share/man/man1/$command.1.gz"
И then less /usr/share/man/man1/$command.1.gz
И else
М echo "No such a command!"
И fi
```

Рис. 4: Скрипт к заданию 2.

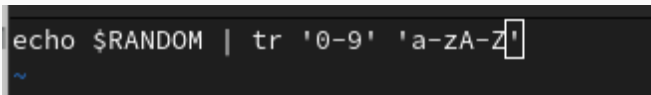
Затем я добавила право на исполнение файла и выполнила его.

A terminal window with a dark background and light gray text. The prompt is [aoaristova@fedora lab12]\$. The commands entered are: vi lab12_2, chmod +x lab12_2, ./lab12_2 -c ls, and a blank line with a cursor. The output of the last command is not visible.

```
1 [aoaristova@fedora lab12]$ vi lab12_2
2 [aoaristova@fedora lab12]$ chmod +x lab12_2
3 [aoaristova@fedora lab12]$ ./lab12_2 -c ls
4 [aoaristova@fedora lab12]$
```

Рис. 5: Результат выполнения скрипта 2.

3. Я написала командный файл, который генерировал случайную последовательность букв латинского алфавита, для этого я использовала встроенную переменную \$RANDOM.

A screenshot of a terminal window with a dark background. The command 'echo \$RANDOM | tr '0-9' 'a-zA-Z'' is entered in white text. A white cursor box is positioned at the end of the command, after the closing single quote. A small blue tilde '~' is visible on the line below the command.

```
echo $RANDOM | tr '0-9' 'a-zA-Z'
```

Рис. 6: Скрипт к заданию 3.

Затем я добавила право на исполнение файла и выполнила его.

```
[aoaristova@fedora lab12]$ vi lab12_3
[aoaristova@fedora lab12]$ chmod +x lab12_3
[aoaristova@fedora lab12]$ ./lab12_3
bejgj
```

Рис. 7: Результат выполнения скрипта 3.

В ходе выполнения лабораторной работы я изучила основы программирования в оболочке ОС UNIX, а также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.