

Лабораторная работы №12 “Программирование в командном процессоре ОС UNIX. Расширенное программирование”

Мажитов М.А.

RUDN University, Moscow, Russian Federation

Цель работы:

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`.

- 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.

1. Написал 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). (рис. 1)

```
#!/bin/bash
LOCKFILE="./lock.file"
exec {fn}>$LOCKFILE

if test -f "$LOCKFILE"
then
    while
        [ 1 = 1 ]
    do
        if flock -n ${fn}
        then
            echo "Файл заблокирован"
            sleep 3
            echo "Файл разблокирован"
            flock -u ${fn}
        else
            echo "Файл заблокирован"
            sleep 3
        fi
    done
fi
```

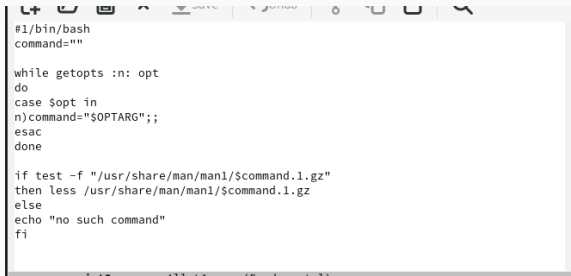
Figure 1: Код 1 скрипта

Запустил скрипт. (рис. 2)

```
[mamazhitov@fedora lab12]$ chmod +x script1
[mamazhitov@fedora lab12]$ ./script1
Файл заблокирован
Файл разблокирован
Файл заблокирован
^C
[mamazhitov@fedora lab12]$
```

Figure 2: Работа скрипта

2. Реализовал команду `man` с помощью командного файла. Командный файл получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`. (рис. 3)

A screenshot of a terminal window with a light gray background and a dark gray title bar. The terminal displays a shell script for the 'man' command. The script starts with a shebang line, initializes a 'command' variable, and enters a loop that processes command-line options. It then checks if a file exists in the '/usr/share/man/man1/' directory and either displays its contents or prints an error message.

```
#!/bin/bash
command=""

while getopts :n: opt
do
case $opt in
n)command="$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "no such command"
fi
```

Figure 3: Код 2 скрипта

Запустил скрипт.(рис. 4)

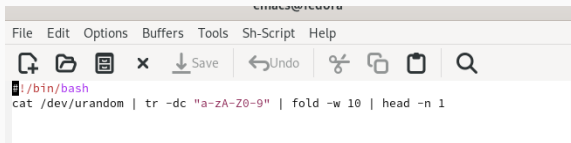
```
BASH_BUILTINS(1)          General Commands Manual          BASH_BUILTINS(1)

ESC[1mNAMEESC[0m
    bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command,
    compgen, complete, compopt, continue, declare, dirs, disown, echo, en-
    able, eval, exec, exit, export, false, fc, fg, getopts, hash, help,
    history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd,
    pwd, read, readonly, return, set, shift, shopt, source, suspend, test,
    times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait -
    bash built-in commands, see ESC[1mbashESC[22m(1)

ESC[1mBASH BUILTIN COMMANDSESC[0m
    Unless otherwise noted, each builtin command documented in this section
    as accepting options preceded by ESC[1m- ESC[22maccepts ESC[1m-- ESC[22mto
    signify the end of the
    options. The ESC[1m-ESC[22m, ESC[1mtrueESC[22m, ESC[1mfalseESC[22m, and
    ESC[1mtestESC[22m/ESC[1m[ ESC[22mbuiltins do not accept options
    and do not treat ESC[1m-- ESC[22mspecially. The ESC[1mexitESC[22m, ESC[1m
    logoutESC[22m, ESC[1mreturnESC[22m, ESC[1mbreakESC[22m, ESC[1mcon-ESC[0m
    ESC[1mtinueESC[22m, ESC[1mletESC[22m, and ESC[1mshift ESC[22mbuiltins a
    ccept and process arguments beginning
    with ESC[1m- ESC[22mwithout requiring ESC[1m--ESC[22m. Other builtins th
    at accept arguments but
    /usr/share/man/man1/cd.1.gz
```

Figure 4: Работа скрипта

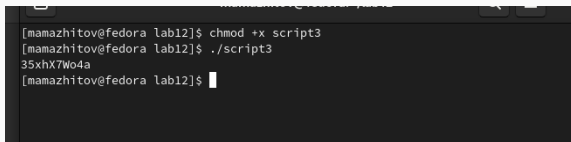
3. Используя встроенную переменную \$RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита.(рис. 5)

A screenshot of a terminal window with a menu bar (File, Edit, Options, Buffers, Tools, Sh-Script, Help) and a toolbar with icons for file operations and editing. The terminal shows a prompt and a command to generate random letters.

```
#!/bin/bash
cat /dev/urandom | tr -dc "a-zA-Z0-9" | fold -w 10 | head -n 1
```

Figure 5: Код 3 скрипта

Запустил скрипт.(рис. 6)

A terminal window with a dark background and light text. The prompt is [mamazhitov@fedora lab12]\$. The first command is chmod +x script3. The second command is ./script3. The output is 35xhX7Wo4a. The prompt is now [mamazhitov@fedora lab12]\$ with a cursor.

```
[mamazhitov@fedora lab12]$ chmod +x script3
[mamazhitov@fedora lab12]$ ./script3
35xhX7Wo4a
[mamazhitov@fedora lab12]$
```

Figure 6: Работа скрипта

Мы научились писать более сложные командные файлы.