

Лабораторная работа No 12. Программирование в командном процессоре ОС UNIX. Расширенное программирование

Leysan R. Abdullina

NEC-2022, 25 May

RUDN University, Moscow, Russian Federation

Лабораторная работа No 12.

Программирование в командном
процессоре ОС UNIX. Расширенное
программирование

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

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; - C-оболочка (или csh)—надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

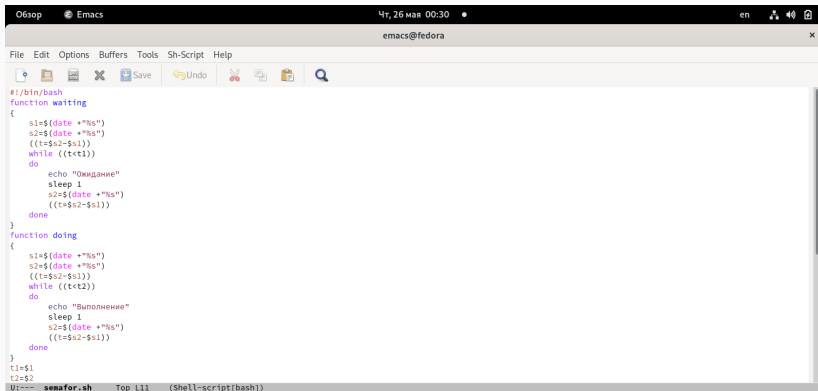
- оболочка Корна (или ksh)—напоминает оболочку C,но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна),в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable 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.

Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t_1 будет 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.

Выполнение работы

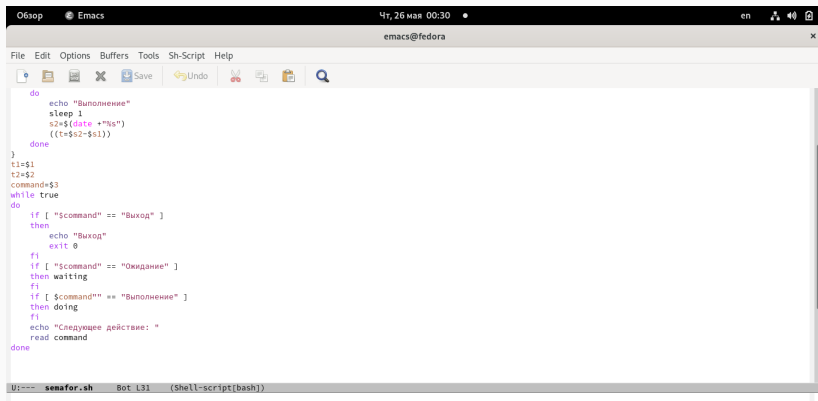
Для этого мы создаем файл `semafor.sh` и пишем программу (скриншоты 1, 2)



```
#!/bin/bash
function waiting
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
function doing
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
t1=1
t2=2
waiting &
doing &
wait
```

Figure 1: Написанная программа

Выполнение работы



The image shows a screenshot of the Emacs text editor. The title bar at the top indicates the window is titled 'Обзор Emacs' and shows the date and time 'Чт, 26 мая 00:30'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The toolbar contains icons for opening files, saving, undo, redo, and search. The main editing area displays a shell script with the following content:

```
do
    echo "Выполнение"
    sleep 1
    s2=$(date +%s)
    ((t=s2-$s1))
done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then
        echo "Выход"
        exit 0
    fi
    if [ "$command" == "Ожидание" ]
    then waiting
    fi
    if [ "$command" == "Выполнение" ]
    then doing
    fi
    echo "Следующее действие: "
    read command
done
```

The status bar at the bottom shows the current file is 'semafor.sh', the user is 'Bot L31', and the shell is '(Shell-script(bash))'.

Figure 2: Написанная программа

Выполнение работы

Не забудем сделать наш файл с программой исполняемым через команду `chmod +x semafor.sh`. После этого вводим команды в консоли `./semafor` и `./semafor 3 5` и смотрим на результат. (скриншот 3)



```
Обзор Терминал Чт, 26 мая 00:30 en
lrabdu1lina@fedora:~/work/study/2021-2022/Операционные системы/os-intro/labs/lab12/progi

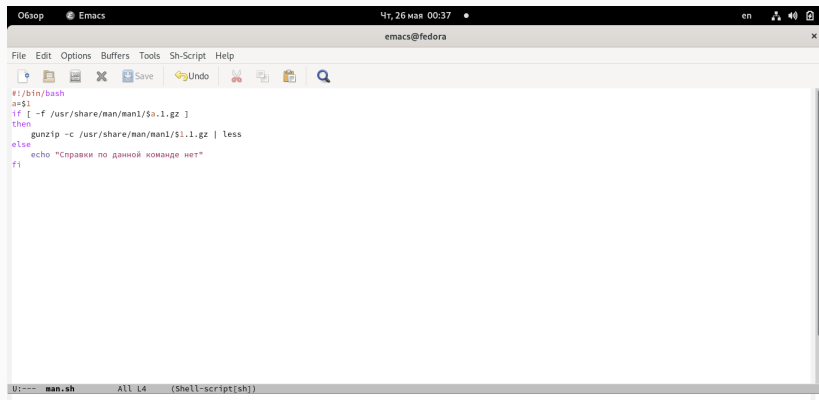
[lrabdu1lina@fedora prog1]$ emacs &
[1] 5455
[lrabdu1lina@fedora prog1]$ ./semafor.sh
Следующее действие:
Ожидание
Следующее действие:
Выполнение
Следующее действие:
Выход
Выход
[lrabdu1lina@fedora prog1]$ ./semafor.sh 3 5
Следующее действие:
Ожидание
Ожидание
Ожидание
Следующее действие:
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие:
Выход
Выход
[lrabdu1lina@fedora prog1]$
```

Figure 3: Проверка - работает успешно

Реализуем команду `man` с помощью командного файла. Изучим 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`.

Для этого мы создаем файл `man.sh` и пишем код (скриншот 4)



```
#!/bin/bash
a=$1
if [ -f /usr/share/man/man1/$a.1.gz ]
then
  gunzip -c /usr/share/man/man1/$1.1.gz | less
else
  echo "Справки по данной команде нет"
fi
```

The screenshot shows the Emacs editor interface. The title bar indicates the file is `man.sh` and the buffer is `All L4 (Shell-script[sh])`. The menu bar includes File, Edit, Options, Buffers, Tools, Sh-Script, and Help. The toolbar contains icons for file operations and search. The main text area contains the shell script code shown above.

Figure 4: Написанная программа

Выполнение работы

Сделаем файл исполняемым через команду `chmod +x man.sh`. И проверим его работу. Пишем команды `./man.sh touch ./man.sh mkdir ./man.sh mksir` (скриншоты 5, 6, 7)

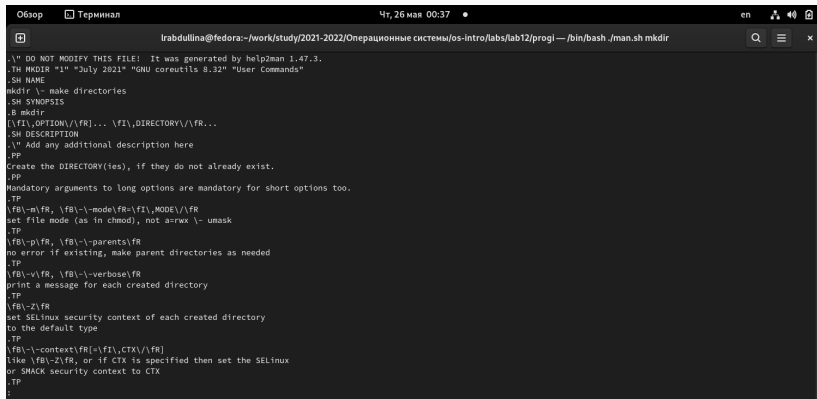
```

Обзор  Терминал  Чт, 26 мая 00:37
irabdu1lina@fedora:~/work/study/2021-2022/Операционные системы/os-intro/labs/lab12/progi — /bin/bash ./man.sh touch
.\\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH TOUCH "1" "July 2021" "GNU coreutils 8.32" "User Commands"
.SH NAME
touch \- change file timestamps
.SH SYNOPSIS
.B touch
[FR], [OPTION]FR]... [fi]FR [fr]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Update the access and modification times of each FILE to the current time.
.PP
A FILE argument that does not exist is created empty, unless FB -cFR or FB -hFR
is supplied.
.PP
A FILE argument string of \- is handled specially and causes touch to
change the times of the file associated with standard output.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
FB -aFR
change only the access time
.TP
FB -cFR, FB -\no\createFR
do not create any files
.TP
FB -dFR, FB -\dateFR=fi,STRINGFR
parse STRING and use it instead of current time
.TP
FB -fFR
(ignored)

```

Figure 5: Проверка - работает успешно

Выполнение работы



```
Обзор Терминал Чт, 26 мая 00:37 en
lrabduullina@fedora:~/work/study/2021-2022/Операционные системы/os-intro/labs/lab12/progi — /bin/bash ./man.sh mkdir

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH MKDIR "1" "July 2021" "GNU coreutils 8.32" "User Commands"
.SH NAME
mkdir \- make directories
.SH SYNOPSIS
.B mkdir
[\fI\OPTION\[/fR]... \fI\,DIRECTORY\[/fR...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Create the DIRECTORY(ies), if they do not already exist.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\m\[/fR, \fB\-\-mode\[/fR=\fI\,MODE\[/fR
set file mode (as in chmod), not a=rwx \- umask
.TP
\fB\p\[/fR, \fB\-\-parents\[/fR
no error if existing, make parent directories as needed
.TP
\fB\v\[/fR, \fB\-\-verbose\[/fR
print a message for each created directory
.TP
\fB\Z\[/fR
set SELinux security context of each created directory
to the default type
.TP
\fB\l-context\[/fR[=\fI\,CTX\[/fR]
like \fB\Z\[/fR, or if CTX is specified then set the SELinux
or SHACK security context to CTX
.TP
:
```

Figure 6: Проверка - работает успешно

Выполнение работы

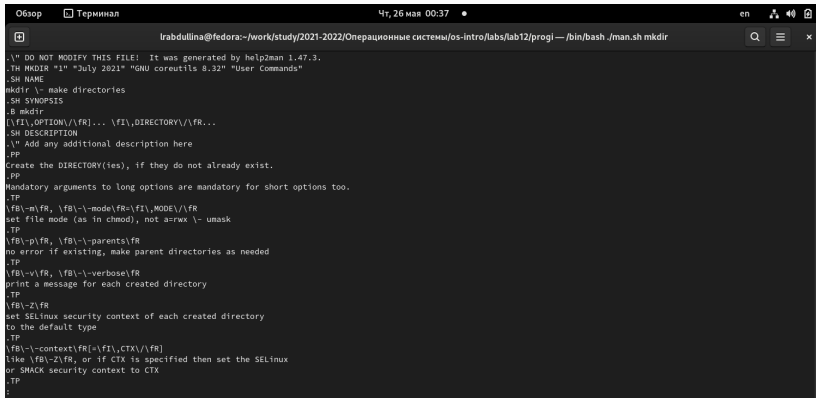
A screenshot of a terminal window. The title bar shows tabs for 'Обзор' and 'Терминал', the date and time 'Чт, 26 мая 00:37', and system icons. The terminal title is 'lrabduullina@fedora:~/work/study/2021-2022/Операционные системы/os-intro/labs/lab12/progi'. The terminal content shows a sequence of commands and outputs: 'Выход', 'Выход', '[lrabduullina@fedora progi]\$./semafor.sh 3 5', 'Следующее действие:', 'Ожидание', 'Ожидание', 'Ожидание', 'Ожидание', 'Следующее действие:', 'Выполнение', 'Выполнение', 'Выполнение', 'Выполнение', 'Выполнение', 'Выполнение', 'Выполнение', 'Следующее действие:', 'Выход', 'Выход', '[lrabduullina@fedora progi]\$ touch man.sh', '[1]+ Завершен emacs', '[lrabduullina@fedora progi]\$ emacs &', '[1] 5551', '[lrabduullina@fedora progi]\$ chmod +x man.sh', '[lrabduullina@fedora progi]\$./man.sh touch', '[lrabduullina@fedora progi]\$./man.sh mksir', 'Справки по данной команде нет', '[lrabduullina@fedora progi]\$./man.sh mkdir', '[lrabduullina@fedora progi]\$./man.sh touch', '[lrabduullina@fedora progi]\$./man.sh mkdir', '[lrabduullina@fedora progi]\$./man.sh mksir', 'Справки по данной команде нет', '[lrabduullina@fedora progi]\$'.

Figure 7: Проверка - работает успешно

3. Используя встроенную переменную `$RANDOM`, напомним командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтем, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до 32767

Выполнение работы

Для начала создадим новый файл random.sh и напомним программу.
(скриншот 8)



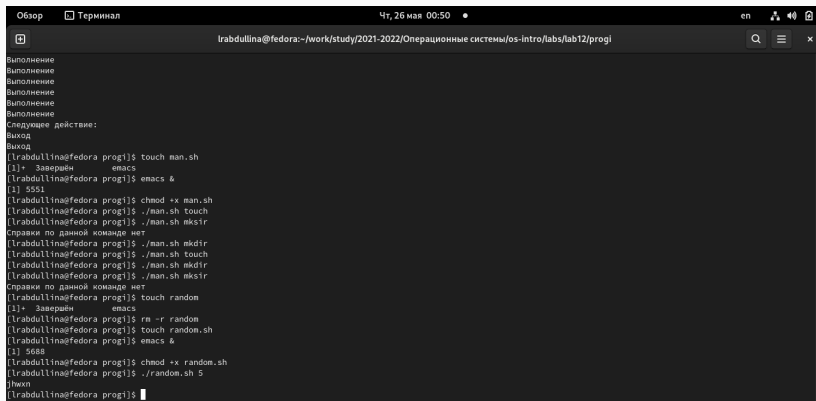
```
Обзор Терминал Чт, 26 мая 00:37 en
lrabduhina@fedora:~/work/study/2021-2022/Операционные системы/os-intro/labs/lab12/progi — /bin/bash .man.sh mkdir

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH MKDIR "1" "July 2021" "GNU coreutils 8.32" "User Commands"
.SH NAME
mkdir \- make directories
.SH SYNOPSIS
.B mkdir
[ \[FI\,OPTION\]\[FR\]... \[FI\,DIRECTORY\]\[FR\]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Create the DIRECTORY(ies), if they do not already exist.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\[/B\[/m\[/FR, \[/B\[/\-mode\[/FR\[/fi\,MODE\]\[/FR
set file mode (as in chmod), not a=rwx \[/- umask
.TP
\[/B\[/p\[/FR, \[/B\[/\-parents\[/FR
no error if existing, make parent directories as needed
.TP
\[/B\[/v\[/FR, \[/B\[/\-verbose\[/FR
print a message for each created directory
.TP
\[/B\[/Z\[/FR
set SELinux security context of each created directory
to the default type
.TP
\[/B\[/\-context\[/FR\[/=fi\,CTX\]\[/FR\]
like \[/B\[/Z\[/FR, or if CTX is specified then set the SELinux
or SMACK security context to CTX
.TP
:
```

Figure 8: Написанная программа

Выполнение работы

Сделаем файл исполняемым через команду `chmod +x random.sh`. И проверим его работу. Напишем случайное число и посмотрим на результат. (скриншот 9)



```
Обзор Терминал Чт, 26 мая 00:50 en [иконки]
lrabdu1lina@fedora:~/work/study/2021-2022/Операционные системы/os-intro/labs/lab12/progi
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие:
Выход
Выход
[lrabdu1lina@fedora progi]$ touch man.sh
[1]+  Завершён  emacs
[lrabdu1lina@fedora progi]$ emacs &
[1] 5551
[lrabdu1lina@fedora progi]$ chmod +x man.sh
[lrabdu1lina@fedora progi]$ ./man.sh touch
[lrabdu1lina@fedora progi]$ ./man.sh mkdir
Справки по данной команде нет
[lrabdu1lina@fedora progi]$ ./man.sh mkdir
[lrabdu1lina@fedora progi]$ ./man.sh touch
[lrabdu1lina@fedora progi]$ ./man.sh mkdir
[lrabdu1lina@fedora progi]$ ./man.sh mkdir
Справки по данной команде нет
[lrabdu1lina@fedora progi]$ touch random
[1]+  Завершён  emacs
[lrabdu1lina@fedora progi]$ rm -r random
[lrabdu1lina@fedora progi]$ touch random.sh
[lrabdu1lina@fedora progi]$ emacs &
[1] 5688
[lrabdu1lina@fedora progi]$ chmod +x random.sh
[lrabdu1lina@fedora progi]$ ./random.sh 5
jhwon
[lrabdu1lina@fedora progi]$
```

Figure 9: Проверка - работает успешно

В ходе лабораторной работы мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.