

Отчёт по лабораторной работе №12

Паращенко Антонина Дмитриевна

Содержание

1	Цель работы	5
2	Ход лабораторной работы	6
2.1	Задание 1	6
2.2	Задание 2	10
2.3	Задание 3	13
3	Вывод	17

Список иллюстраций

2.1	Консоль	6
2.2	Скрипт	7
2.3	Работа программы	8
2.4	Скрипт	9
2.5	Работа программы	10
2.6	Содержимое	11
2.7	Скрипт кода	11
2.8	Запуск файла	12
2.9	mkdir	12
2.10	Запуск файла	12
2.11	ls	13
2.12	Запуск файла	13
2.13	Скрипт	15
2.14	Скрипт	16

Список таблиц

1 Цель работы

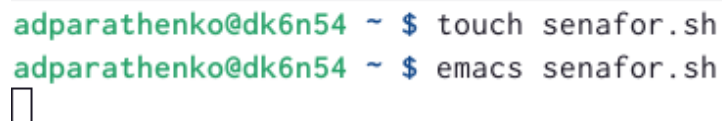
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Ход лабораторной работы

2.1 Задание 1

Написать командный файл, реализующий упрощённый механизм семафоров

1) Создаём файл `senafor.sh` и открываем его в редакторе `emacs` (рис. 2.1)



```
adparathenko@dk6n54 ~ $ touch senafor.sh
adparathenko@dk6n54 ~ $ emacs senafor.sh
```

□

Рис. 2.1: Консоль

2) Пишем скрипт командного файла (рис. 2.2)

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

Рис. 2.2: Скрипт

3) Даём права на исполнение файла и запускаем командный файл (рис. 2.3)

```
adparathenko@dk6n54 ~ $ chmod +x senafor.sh
adparathenko@dk6n54 ~ $ ./semafor.sh 3 5
bash: ./semafor.sh: Нет такого файла или каталога
adparathenko@dk6n54 ~ $ ./senafor.sh 3 5
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
adparathenko@dk6n54 ~ $ ./senafor.sh 1 4
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
adparathenko@dk6n54 ~ $ █
```

Рис. 2.3: Работа программы

- 4) Дорабатываем программу так, чтобы имелась возможность взаимодействия трёх и более процессов. (рис. 2.4)


```
#!/bin/bash
function waiting
{
s1=$(date +%s")
s2=$(date +%s")
((t=$s2-$s1))
while ((t < t1))
do
    echo "Waiting"
    sleep 1
    s2=$(date +%s")
    ((t=$s2-$s1))
done
}
function doing
{
s1=$(date +%s")
s2=$(date +%s")
((t=$s2-$s1))
while ((t < t2))
do
    echo "Doing"
    sleep 1
    s2=$(date +%s")
    ((t=$s2-$s1))
done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Exit" ]
    then
        echo "Exit"
        exit 0
    fi
    if [ "$command" == "Wait" ]
    then waiting
    fi
    if [ "$command" == "Do" ]
    then doing
    fi
    echo "Next command"
    read command
done
```

Рис. 2.4: Скрипт

4) Запускаем программу (рис. 2.5)

```
adparathenko@dk6n54 ~ $ ./senafor.sh 2 4
Next command
Doing
Next command
do
Next command
Do
Doing
Doing
Doing
Doing
Next command
Wait
Waiting
Waiting
Next command
Exit
Exit
adparathenko@dk6n54 ~ $ ./senafor.sh 2 4 > Waiting > /dev/tty54
bash: /dev/tty54: Отказано в доступе
adparathenko@dk6n54 ~ $ █
```

Рис. 2.5: Работа программы

2.2 Задание 2

Реализовать команду `man` с помощью командного файла. 1) Изучаем содержимое каталога `/usr/share/man/man1`. (рис. 2.6)

```

adparathenko@dk6n54 /usr/share/man/man1 $ ls
411toppm.1.bz2      npm-bugs.1.bz2
7z.1.bz2            npm-cache.1.bz2
7za.1.bz2           npm-ci.1.bz2
7zr.1.bz2           npm-completion.1.bz2
a2ps.1.bz2          npm-config.1.bz2
a2x.1.bz2           npm-dedupe.1.bz2
a52dec.1.bz2        npm-deprecate.1.bz2
aacplusenc.1.bz2    npm-diff.1.bz2
ab.1.bz2            npm-dist-tag.1.bz2
aclocal-1.12.1.bz2  npm-docs.1.bz2
aclocal-1.13.1.bz2  npm-doctor.1.bz2
aclocal-1.14.1.bz2  npm-edit.1.bz2
aclocal-1.15.1.bz2  npm-exec.1.bz2
aclocal-1.16.1.bz2  npm-explain.1.bz2
aconnect.1.bz2      npm-explore.1.bz2
acyclic.1.bz2       npm-find-dupes.1.bz2
adddebug.1.bz2      npm-fund.1.bz2
addedge.1.bz2       npm-help.1.bz2
addftinfo.1.bz2     npm-help-search.1.bz2
addrinfo.1.bz2      npm-hook.1.bz2
advdef.1.bz2        npm-init.1.bz2
advpng.1.bz2        npm-install.1.bz2
advzip.1.bz2        npm-install-ci-test.1.bz2
afm2pl.1.bz2        npm-install-test.1.bz2
afm2tfm.1.bz2       npm-link.1.bz2
afmtodit.1.bz2      npm-logout.1.bz2
afs.1.bz2           npm-ls.1.bz2
afs_compile_et.1.bz2 npm-org.1.bz2
afsmonitor.1.bz2    npm-outdated.1.bz2
agentxtrap.1.bz2    npm-owner.1.bz2
aklog.1.bz2         npm-pack.1.bz2
al.1.bz2            npm-ping.1.bz2
alacarte.1.bz2      npm-pkg.1.bz2
aleph.1.bz2         npm-prefix.1.bz2
aliastorle.1.bz2   npm-profile.1.bz2
align_image_stack.1.bz2 npm-prune.1.bz2
allcm.1.bz2         npm-publish.1.bz2
allec.1.bz2         npm-rebuild.1.bz2
allneeded.1.bz2     npm-repo.1.bz2
alsactl.1.bz2       npm-restart.1.bz2
alsa-info.sh.1.bz2  npm-root.1.bz2
alsaloop.1.bz2      npm-run-script.1.bz2
                    npm-search.1.bz2

```

Рис. 2.6: Содержимое

- 2) Создаём файл `man.sh` и открываем его в редакторе `emacs`. Пишем скрипт командного файла (рис. 2.7)

```

#!/bin/bash
x=$1
if [ -f /usr/share/man/man1/$x.1.bz2 ]
then
    bunzip2 -c /usr/share/man/man1/$1.1.bz2 | less
else
    echo "There is no information about it"
fi

```

Рис. 2.7: Скрипт кода

- 3) Даём право на исполнение и проверяем работу командного файла с командами `mkdir` и `ls` (рис. 2.8) - (рис. 2.11)

```
adparathenko@dk6n54 ~ $ chmod +x man.sh
adparathenko@dk6n54 ~ $ ./man.sh mkdir
adparathenko@dk6n54 ~ $
```

Рис. 2.8: Запуск файла

```
." DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH MKDIR "1" "March 2020" "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\--context\\fR[=\fI\,CTX\\fR]
like \fB\-Z\\fR, or if CTX is specified then set the SELinux
or SMACK security context to CTX
.TP
\fB\--help\\fR
display this help and exit
.TP
\fB\--version\\fR
output version information and exit
.SH AUTHOR
Written by David MacKenzie.
.SH "REPORTING BUGS"
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
.br
Report any translation bugs to <https://translationproject.org/team/>
.SH "SEE ALSO"
mkdir(2)
lines 1-44
```

Рис. 2.9: mkdir

```
adparathenko@dk6n54 ~ $ ./man.sh ls
adparathenko@dk6n54 ~ $
```

Рис. 2.10: Запуск файла

```

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH LS "1" "March 2020" "GNU coreutils 8.32" "User Commands"
.SH NAME
ls \- list directory contents
.SH SYNOPSIS
.B ls
[\fI\,OPTION\[\fR]... [\fI\,FILE\[\fR]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB\-cftuvSUX\[\fR nor \fB\-\-sort\[\fR is specified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\-a\[\fR, \fB\-\-all\[\fR
do not ignore entries starting with .
.TP
\fB\-A\[\fR, \fB\-\-almost-\-all\[\fR
do not list implied . and ..
.TP
\fB\-\-author\[\fR
with \fB\-l\[\fR, print the author of each file
.TP
\fB\-b\[\fR, \fB\-\-escape\[\fR
print C\-style escapes for nongraphic characters
.TP
\fB\-\-block-size\[\fR=\fI\,SIZE\[\fR
with \fB\-l\[\fR, scale sizes by SIZE when printing them;
e.g., '\-\-block-size=M'; see SIZE format below
.TP
\fB\-B\[\fR, \fB\-\-ignore-\-backups\[\fR
do not list implied entries ending with ~
.TP
\fB\-c\[\fR
with \fB\-lt\[\fR: sort by, and show, ctime (time of last
modification of file status information);
with \fB\-l\[\fR: show ctime and sort by name;
otherwise: sort by ctime, newest first
.TP
\fB\-C\[\fR
list entries by columns
.TP
\fB\-\-color\[\fR[=\fI\,WHEN\[\fR]
lines 1-44

```

Рис. 2.11: ls

4) Проверяем работу файла с несуществующей командой rted (рис. 2.12)

```

adparathenko@dk6n54 ~ $ ./man.sh rted
There is no information about it
adparathenko@dk6n54 ~ $ █

```

Рис. 2.12: Запуск файла

2.3 Задание 3

Используя встроенную переменную \$RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. 1) Созда-

ём файл `map.sh` и открываем его в редакторе `emacs`. Пишем скрипт командного файла (рис. 2.13)

```
#!/bin/bash
k=$1
for ((i=0; i<$k; i++ ))
do
    (( char=$RANDOM%26+1))
    case $char in
        1) echo -n a;;
        2) echo -n b;;
        3) echo -n c;;
        4) echo -n d;;
        5) echo -n e;;
        6)echo -n f;;
        7)echo -n g;;
        8)echo -n h;;
        9)echo -n i;;
        10)echo -n j;;
        11)echo -n k;;
        12)echo -n l;;
        13)echo -n m;;
        14)echo -n n;;
        15)echo -n o;;
        16)echo -n p;;
        17)echo -n q;;
        18)echo -n r;;
        19)echo -n s;;
        20)echo -n t;;
        21)echo -n u;;
        22)echo -n v;;
        23)echo -n w;;
        24)echo -n x;;
        25)echo -n y;;
        26)echo -n z
    esac
done
echo
```

Рис. 2.13: Скрипт

2) Запускаем файл и проверяем правильность работы (рис. 2.14)

```
adparathenko@dk6n54 ~ $ ./random.sh 3  
wfu  
adparathenko@dk6n54 ~ $ ./random.sh 20  
ixkguaoctbwgabpimshs  
adparathenko@dk6n54 ~ $ █
```

Рис. 2.14: Скрипт

3 Вывод

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