

Презентация по лабораторной работе №12

Подъярова Ксения Витальевна

Российский Университет Дружбы Народов

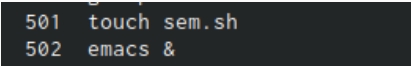
Цель работы

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

Выполнение лабораторной работы

Задание 1

- 1) Написала 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). Для данной задачи я создала файл `sem.sh` (рис. 1) и написала соответствующий скрипт (рис. 2).



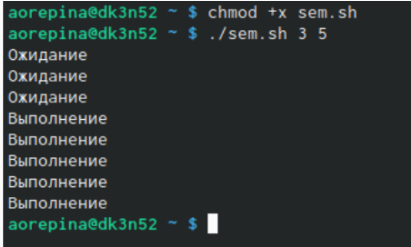
```
501 touch sem.sh
502 emacs &
```

Figure 1: Создание файла

```
#!/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
```

Figure 2: Скрипт

- 2) Далее я проверила работу написанного скрипта (`./sem.sh 4 7`), предварительно предоставив файлу право на исполнение (`chmod +x sem.sh`). (рис. 3). Скрипт работает корректно



```
aorepina@dk3n52 ~ $ chmod +x sem.sh
aorepina@dk3n52 ~ $ ./sem.sh 3 5
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
aorepina@dk3n52 ~ $
```

Figure 3: Проверка работы

- 3) После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и прверила его работу (например команда `./sem.sh 2 3 Ожидание > /dev/pts/1 &`) (рис. 4) (рис. 5). После проверила работу скрипта и увидела, что мне было отказано в доступе (рис. 6)

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

Figure 4: Скрипт


```
while ((t<t2))
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 ozhidanie
    fi
```

Figure 5: Скрипт

```
./sem.sh 2 3 Ожидание > /dev/pts/1 &  
  
bash: /dev/pts/1: Отказано в доступе  
  
./sem.sh 2 3 Ожидание > /dev/pts/1  
./sem.sh 2 5 Выполнение > /dev/pts/2 &
```

Figure 6: Проверка работы

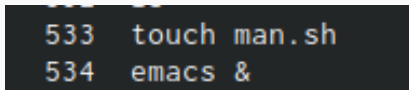
Задание 2

2. 1) Реализовала команду `man` с помощью командного файла. Изучила содержимое каталога `/usr/share/man/man1` (рис. 7). 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`

[illegible]

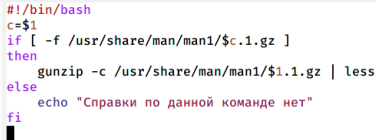
Figure 7: Содержимое каталога /usr/share/man/man1

- 2) Для данной задачи я создала файл man.sh (рис. 8) и написала соответствующий скрипт (рис. 9)

A terminal window with a dark background. Two lines of text are visible: '533 touch man.sh' and '534 emacs &'. The text is in a light blue or cyan color.

```
533 touch man.sh
534 emacs &
```

Figure 8: Создание файла

A terminal window with a light gray background. It shows the content of a shell script. The text is in a monospaced font with syntax highlighting: comments are purple, variables are blue, and other text is black. The script checks for a file in /usr/share/man/man1/ and either gunzips it or prints a message.

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

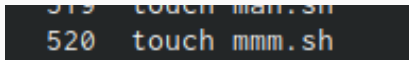
Figure 9: Скрипт

- 3) Далее я проверила работу написанного скрипта (`./man.sh ls` и `./man.sh mkdir`), предварительно добавив право на исполнение файла (`chmod +x man.sh`) (рис. 10). Скрипт работает корректно.

```
aorepina@dk3n52 ~ $ chmod +x man.sh
aorepina@dk3n52 ~ $ ./man.sh ls
Справки по данной команде нет
aorepina@dk3n52 ~ $ ./man.sh mkdir
Справки по данной команде нет
aorepina@dk3n52 ~ $
```

Figure 10: Проверка работы

3. 1) Используя встроенную переменную `$RANDOM`, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Учла, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до 32767. Для данной задачи я создала файл `mmm.sh` (рис. 11) и написала соответствующий скрипт (рис. 12)

A terminal window with a dark background. The prompt is a root symbol. The command `touch mmm.sh` is entered and executed. The output shows the file permissions `-rw-r--r--` and the file name `mmm.sh` in a light blue color.

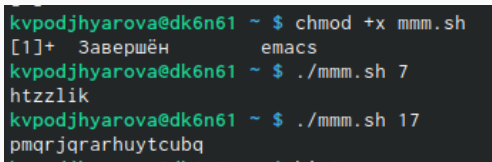
```
root@kali:~# touch mmm.sh
-rw-r--r-- 1 root root 0 Nov 17 12:54 mmm.sh
```

Figure 11: Создание файла

```
#!/bin/bash
k=$1
for (( i=0; i<$k; i++ ))
do
    (( char=$((RANDOM%26+1)) ))
    case $char in
        1) echo -n x;; 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
```

Figure 12: Скрипт

- 2) Далее я проверила работу написанного скрипта (`./random.sh 7; 17`), предварительно добавив право на исполнение файла (рис. 12). Скрипт работает корректно

A terminal window with a dark background and green text. The prompt is 'kvpodjhyarova@dk6n61 ~ \$'. The first command is 'chmod +x mmm.sh', followed by '[1]+ Завершён emacs'. The second command is './mmm.sh 7', which outputs 'htzzlik'. The third command is './mmm.sh 17', which outputs 'pmqrjqrarhuycubq'.

```
kvpodjhyarova@dk6n61 ~ $ chmod +x mmm.sh
[1]+  Завершён      emacs
kvpodjhyarova@dk6n61 ~ $ ./mmm.sh 7
htzzlik
kvpodjhyarova@dk6n61 ~ $ ./mmm.sh 17
pmqrjqrarhuycubq
```

Figure 13: Проверка работы

Выводы

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