

Презентация по лабораторной работе №12

Шмырин Михаил Сергеевич

Российский Университет Дружбы Народов

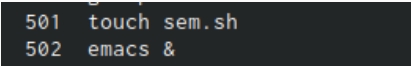
Цель работы

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) 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). Для данной задачи я создал файл `sem.sh` (рис. 1) и написал 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 проверил работу написанного скрипта (`./sem.sh 4 7`), предварительно предоставив файлу право на исполнение (`chmod +x sem.sh`). Скрипт работает корректно

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

```
#!/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 3: 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 4: 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 5: Проверка работы

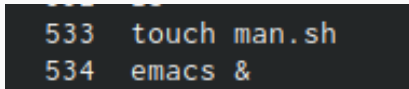
Задание 2

2. 1) Реализовал команду `man` с помощью командного файла. Изучил содержимое каталога `/usr/share/man/man1` (рис. 6). 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]

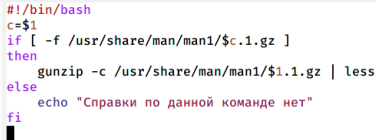
Figure 6: Содержимое каталога /usr/share/man/man1

- 2) Для данной задачи я создал файл man.sh (рис. 7) и написал соответствующий скрипт (рис. 8)

A terminal window with a dark background. The prompt is root@kali:~#. Two lines of text are shown: 533 touch man.sh and 534 emacs &. The text is in a light blue/cyan color.

```
root@kali:~# 533 touch man.sh
root@kali:~# 534 emacs &
```

Figure 7: Создание файла

A screenshot of a text editor showing the content of the man.sh script. The text is color-coded: comments in grey, variables in blue, and control structures in red. The script checks for a file in /usr/share/man/man1/ and either gunzips it and pipes it to less, or prints a message in Russian.

```
#!/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 8: Скрипт

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

3. 1) Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Учла, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. Для данной задачи я создал файл mmm.sh (рис. 9) и написал соответствующий скрипт (рис. 10)

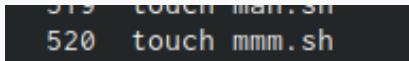


Figure 9: Создание файла

```
#!/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 10: Скрипт

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

Выводы

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