

Презентация к лабораторной работе №13

Старков Никита Алексеевич

¹RUDN 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) Напишем 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.

Создаем файл `sem.sh` и открываем `emacs`.

```
nastarkov@dk6n55 ~ $ touch sem.sh
nastarkov@dk6n55 ~ $ emacs &
```

Пишем скрипт, удовлетворяющий условиям задачи

```
#!/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 №1

Проверяем работу написанного скрипта, предварительно открыв доступ на исполнение файла

```
nastarkov@dk6n55 ~ $ ./sem1.sh 4 7
```

Ожидание

Ожидание

Ожидание

Ожидание

Выполнение

Выполнение

Выполнение

Выполнение

Выполнение

Выполнение

Выполнение

Figure 3: Проверка работы скрипта

Дорабатываем программу в соответствии с условием задачи

```
#!/bin/bash
function ogidania
{
    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
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then
        echo "Выход"
        exit 0
    fi
    if [ "$command" == "Выполнение" ]
    then
        ogidania
    fi
    if [ "$command" == "Выполнение" ]
    then
        vipolnenie
    fi
    echo "Следующее действие: "
    read command
done
```

Figure 4: Доработанный скрипт №1

Проверяем работу

```
rootarkov@okd55 ~ $ chmod +x aom.sh
rootarkov@okd55 ~ $ ./aom.sh 2 3 Ожидание > /dev/pts/1 &
[2] 18934
rootarkov@okd55 ~ $ bash: /dev/pts/1: Отказано в доступе

[2]+  Выход 1 ./aom.sh 2 3 Ожидание > /dev/pts/1
rootarkov@okd55 ~ $ ./aom.sh 2 3 Ожидание > /dev/pts/2 &
[2] 19315
rootarkov@okd55 ~ $ bash: /dev/pts/2: Отказано в доступе

[2]+  Выход 1 ./aom.sh 2 3 Ожидание > /dev/pts/2
rootarkov@okd55 ~ $ ./aom.sh 2 5 Ожидание > /dev/pts/2 &
[2] 19344
rootarkov@okd55 ~ $ bash: /dev/pts/2: Отказано в доступе

[2]+  Выход 1 ./aom.sh 2 5 Ожидание > /dev/pts/2
```

Figure 5: Проверка работы скрипта

2) Реализуем команду `man` с помощью командного файла. Изучаем 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`.

[illegible]

Создаем файл man.sh и пишем скрипт, удовлетворяющий условиям задачи

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

Figure 7: Скрипт №2

Проверяем работу скрипта, предварительно открыв доступ на исполнение файла

```
nastarkov@dk6n55 ~ $ chmod +x man.sh
nastarkov@dk6n55 ~ $ ./man.sh ls
Справки по данной команде нет
nastarkov@dk6n55 ~ $ ./man.sh mkdir
Справки по данной команде нет
```

Figure 8: Проверка работы скрипта

3)Используя встроенную переменную \$RANDOM, напомним командный файл, генерирующий случайную последовательность букв латинского алфавита.

Создаем файл random.sh и пишем скрипт, удовлетворяющий условиям задачи

```
#!/bin/sh
n=1
while [ $n -le 10 ]; do
    ((RANDOM=$RANDOM%26+1))
    echo -n $(cat /dev/urandom | tr -dc 'a-z' | fold -w 1 | head -n 1 | tr -d '\n')
    n=$((n+1))
done
echo
```

Figure 9: Скрипт №3

Далее проверяем работу написанного скрипта, предварительно открыв право на исполнение.

```
nastarkov@okin55 ~$ ./random.sh 10  
weakqofic  
nastarkov@okin55 ~$ ./random.sh 5  
wyyle
```

Figure 10: Проверка работы скрипта

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