Отчет по лабораторной работе №12

Алмазова Елизавета Андреевна

28.05.2022

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

Цель работы и задание

Цель данной лабораторной работы - изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Задание:

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об

Ход работы

1. Написала командный файл semaphor.sh, реализующий упрощённый механизм семафоров. Командный файл, получая в качестве аргумента t1 и t2, в течение некоторого времени t1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени t2, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом) (рис.1).

```
#!/bin/bash

t1=$1                                   #time of waiting
t2=$2                                   #time of using resource
s1=$(date +"%s")                        #s1, s2 - timers
s2=$(date +"%s")
t=$s2-$s1                               #passed time
while ((t < t1))
do
        echo "Waiting"                  #printing waiting
        sleep 1
        s2=$(date +"%s")
        t=$s2-$s1
done
s1=$(date +"%s")
s2=$(date +"%s")
t=$s2-$s1                               #restart time
while ((t+t1 < t2))
do
        echo "Using resource"           #printing using
        sleep 1
        s2=$(date +"%s")
        t=$s2-$s1
done

```

Рисунок 1 - semaphor.sh.

2. Проверила работу файла. Запустила командный файл semaphor.sh в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (> /dev/tty1), в котором также запущен файл semaphorprivileged.sh в привилегированном режиме (рис.2).

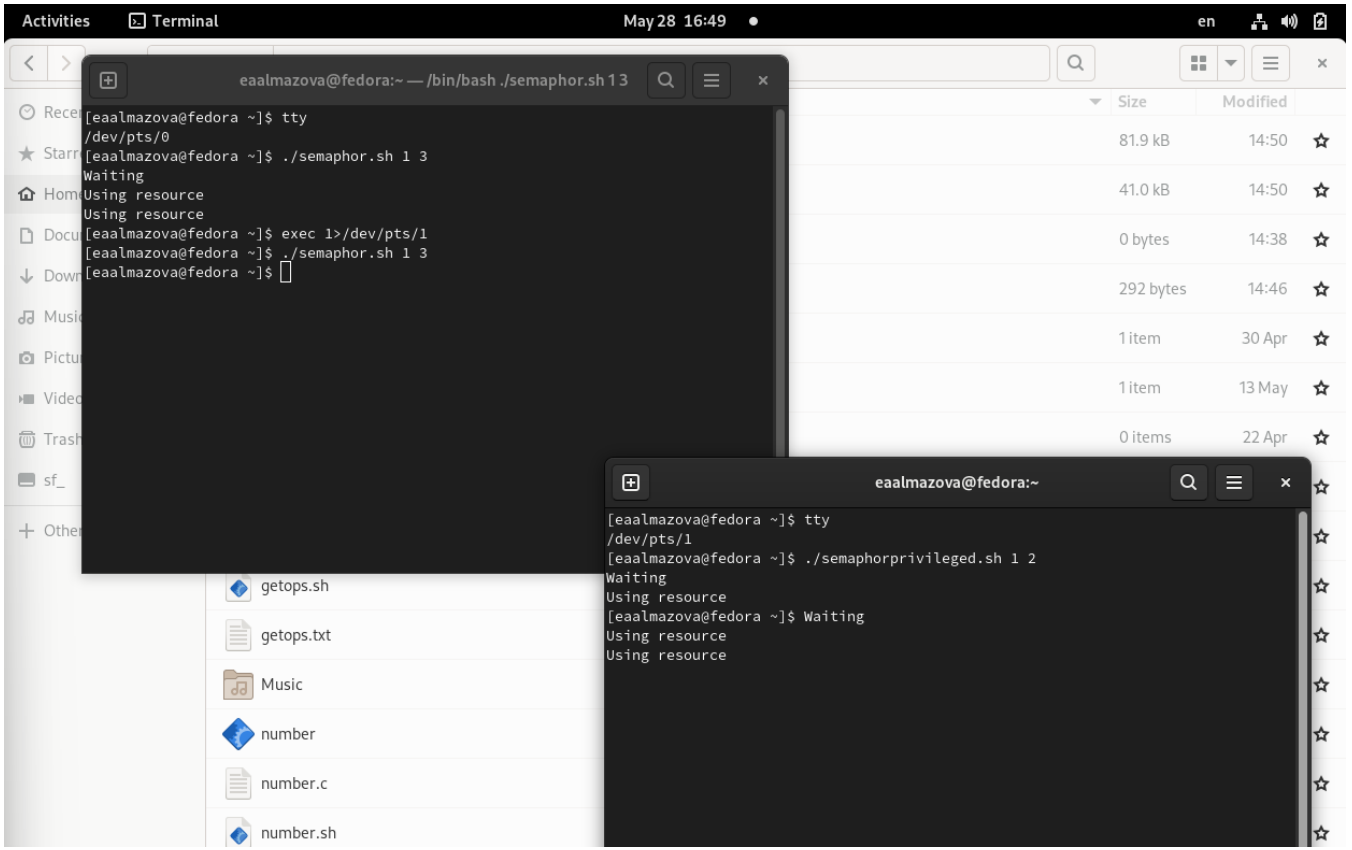

Рисунок 2 - Проверка работы файла и перенаправление вывода.

3. Доработала программу semaphor2.sh так, чтобы имелась возможность взаимодействия трёх и более процессов (рис.3). Проверила работу программы (рис.4).

```
#!/bin/bash

t1=$1                                   #time to wait
t2=$2                                   #time to use resource
N=$3                                    #number of terminals
i=0                                     #iterator of terminals

while (( i < N )); do
        exec 1>/dev/pts/$i              #changing output to terminal #i
        s1=$(date +"%s")
        s2=$(date +"%s")
        t=$s2-$s1                       #setting time
        while (( t < t1 )); do
                echo "$i waiting"       #terminal #i waiting
                sleep 1
                s2=$(date +"%s")
                t=$s2-$s1
        done
        s1=$(date +"%s")
        s2=$(date +"%s")
        t=$s2-$s1+$t1                   #resetting time
        while (( t < t2 )); do
                echo "$i using resource"        #terminal #i using resource
                sleep 1
                s2=$(date +"%s")
                t=$s2-$s1
        done
        i=$(($i+1))                     #changing iterator
done
```

Рисунок 3 - semaphor2.sh.

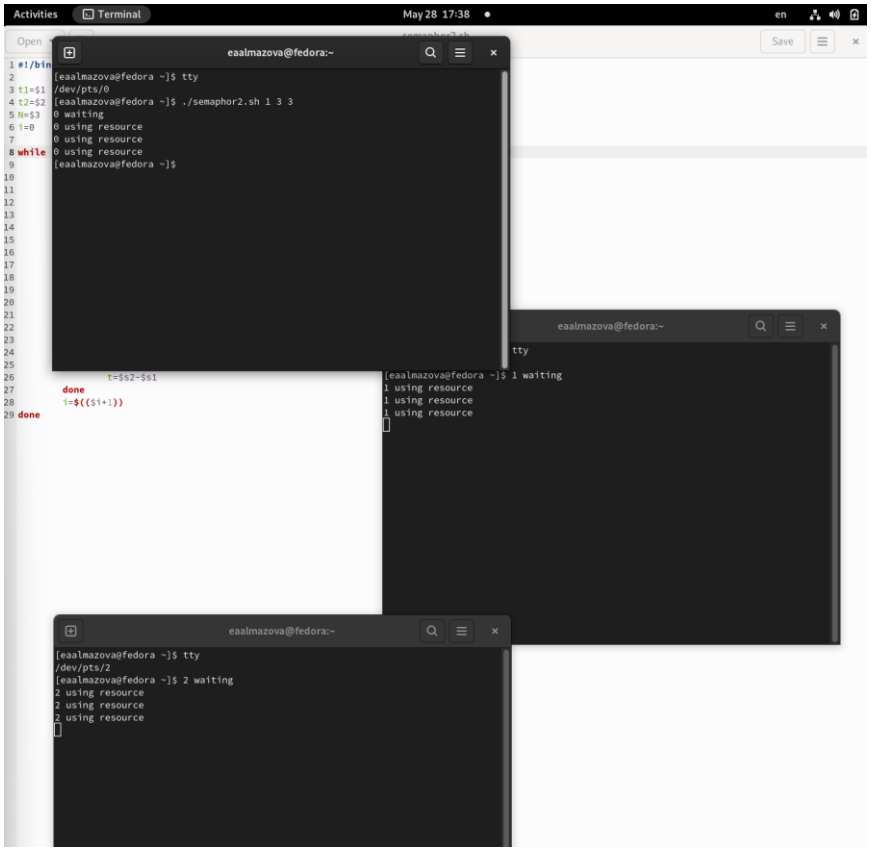

Рисунок 4 - Проверка работы файла.

4. Реализовала команду man с помощью командного файла man.sh. Изучила 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 получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде или сообщение об отсутствии справки, если

```
#!/bin/bash

command=$1                                              #command name
if [ -f /usr/share/man/man1/$command.1.gz ]             #looking for manual
then
        gunzip -c /usr/share/man/man1/$command.1.gz | less      #unarchivating if manual exists
else
        echo "There is no manual"                       #message if there is no manual
fi
```

Рисунок 5 - man.sh.

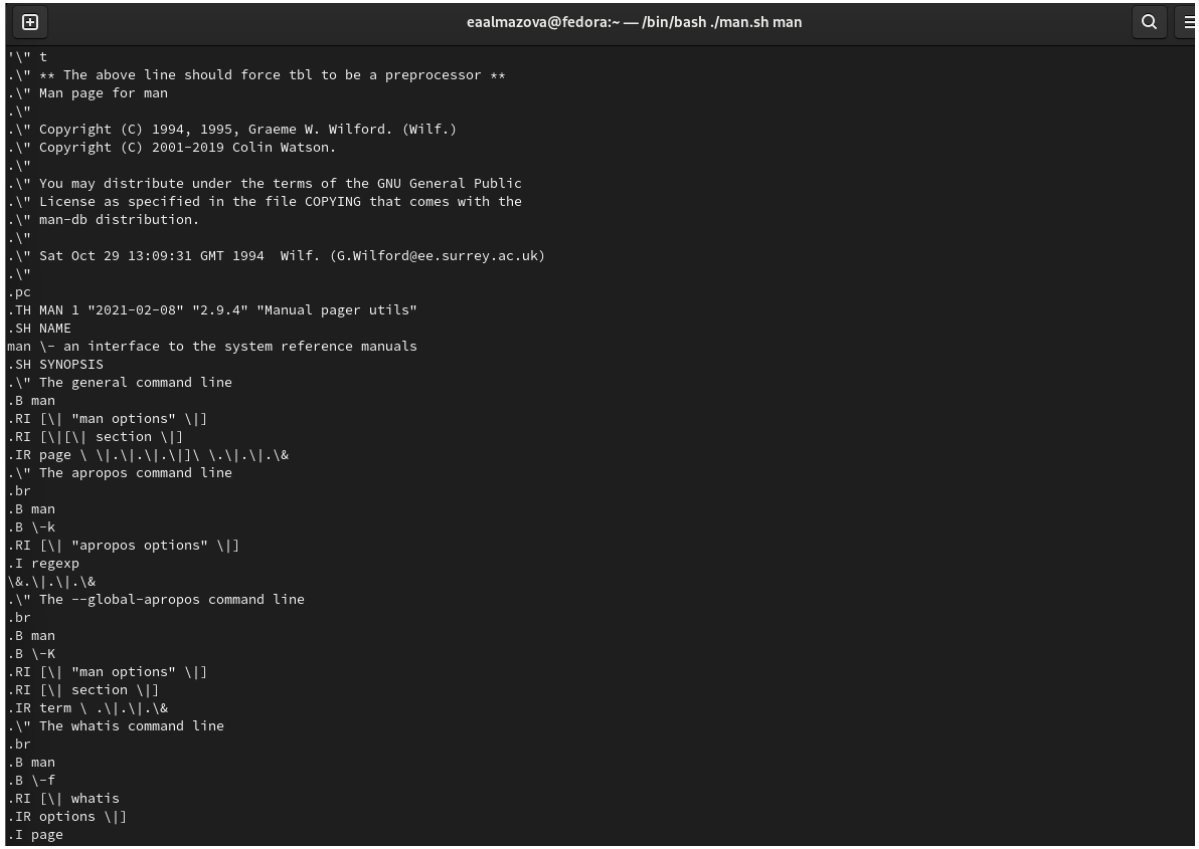

Рисунок 6 - Проверка работы файла.

5. Используя встроенную переменную $RANDOM, написала командный файл random.sh, генерирующий случайную последовательность букв латинского алфавита, получая в качестве аргумента ее длину (рис.7). Проверила работу файла (рис.8).

```
#!/bin/bash

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

Рисунок 7 - random.sh.

```
[eaalmazova@fedora ~]$ ./random.sh 11
fwcjbsdpvl
[eaalmazova@fedora ~]$ ./random.sh 3
frr
```

Рисунок 8 - Проверка работы файла.

Выводы

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

Спасибо за внимание!