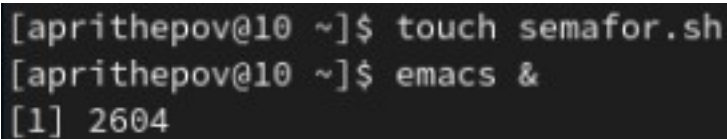


12 Лабораторная Работа (презентация)

Прищепов Александр НПИМ-03-21 ## Введение: - Цель работы: Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов. ## Ход Работы: 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).

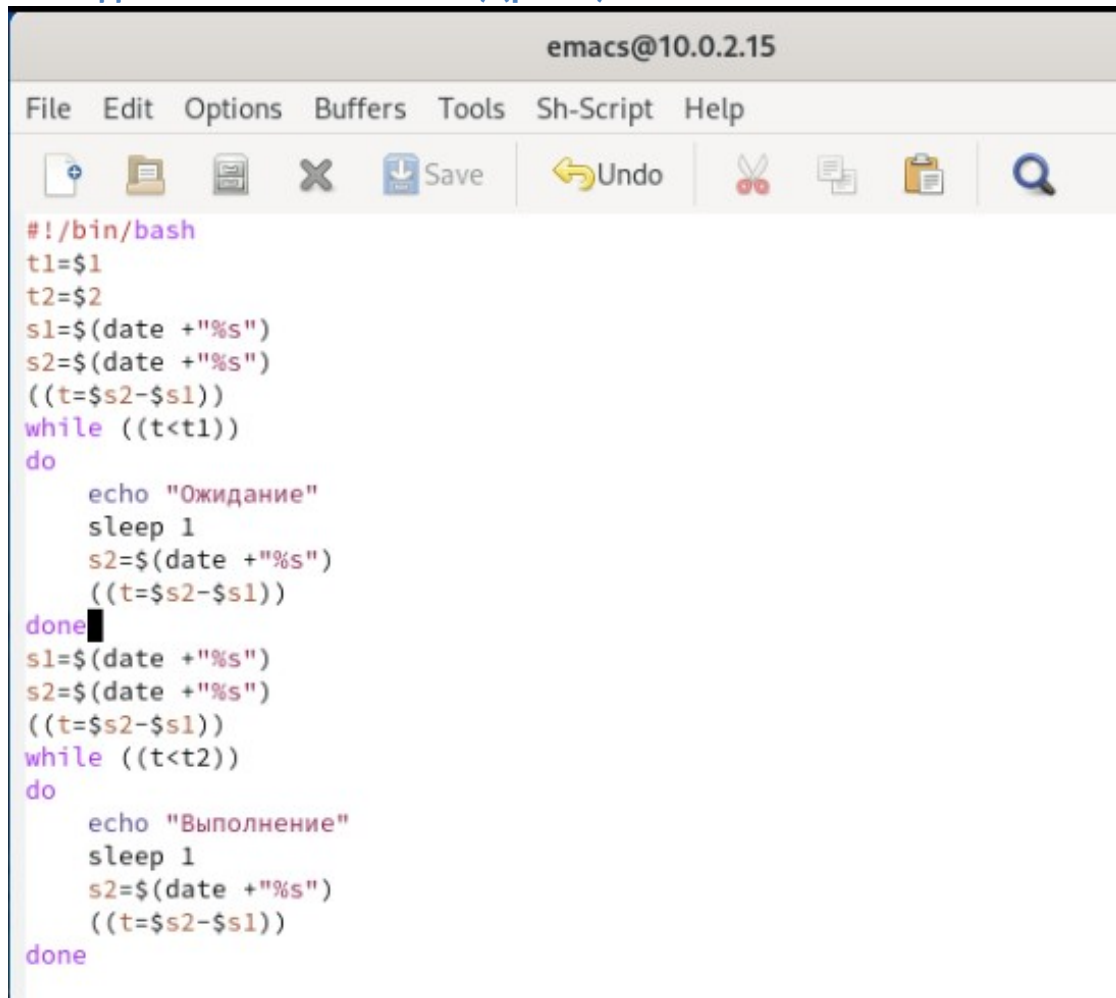
Для выполнения данной задачи создадим файл `semafor.sh` и откроем его в `emacs` (рис.1).



```
[aprithepov@10 ~]$ touch semafor.sh
[aprithepov@10 ~]$ emacs &
[1] 2604
```

изображение

В файле напомним соответствующий скрипт (рис.2) и проверим его работу (команда `./semafor.sh 2 4`), предварительно добавив права на выполнение (команда `chmod +x semafor.sh`) (рис.3).

The image shows a screenshot of the Emacs text editor. The title bar at the top reads "emacs@10.0.2.15". Below the title bar is a menu bar with "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". Under the menu bar is a toolbar with icons for file operations (new, open, save, close, save all) and editing (undo, redo, copy, paste, search). The main text area contains a shell script for a semaphore simulation. The script starts with a shebang line, assigns arguments to variables, calculates a delay, and then enters a loop where it prints "Ожидание" (Waiting) and sleeps for 1 second. After the loop, it prints "Выполнение" (Execution) and sleeps for 1 second. The script ends with a "done" keyword.

```
#!/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
```

изображение

```
[aprithepov@10 ~]$ chmod +x semafor.sh
[aprithepov@10 ~]$ ./semafor.sh 2 4
./semafor.sh: строка 6: синтаксическая ошибка рядом с неожиданным маркером «(»
./semafor.sh: строка 6: `((t=$s2-$s1))'
[aprithepov@10 ~]$ chmod +x semafor.sh
[aprithepov@10 ~]$ ./semafor.sh 2 4
./semafor.sh: строка 6: синтаксическая ошибка рядом с неожиданным маркером «(»
./semafor.sh: строка 6: `((t=$s2-$s1))'
[aprithepov@10 ~]$ ./semafor.sh 2 4
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
```

изображение

Затем изменим скрипт так, чтобы можно было запускать командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (рис.4).

File Edit Options Buffers Tools Sh-Script Help



```
#!/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
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then
        echo "Выход"
        exit 0
    fi
    if [ "$command" == "Выход" ]
    then ozhidanie
    fi
    if [ "$command" == "Выполнение" ]
    then vipolnenie
    fi
    echo "Следующее действие: "
    read command
done
```

U:~-- semafor.sh All L36 (Shell-script[sh])

изображение

Проверим его работу (например, команда `./semafor.sh 2 4 Ожидание > /dev/pts/1`) и увидим, что нам отказано в доступе (рис.6). Но при этом скрипт работает корректно (рис.7) при вводе команды `./semafor.sh 2 4 Ожидание`.

```
[aprithepov@10 ~]$ ./semafor.sh 2 4 Ожидание > /dev/pts/1 &
[2] 3147
bash: /dev/pts/1: Отказано в доступе
[2]+  Выход 1          ./semafor.sh 2 4 Ожидание > /dev/pts/1
[aprithepov@10 ~]$ ./semafor.sh 2 4 Ожидание > /dev/pts/2 &
[2] 3166
bash: /dev/pts/2: Отказано в доступе
[2]+  Выход 1          ./semafor.sh 2 4 Ожидание > /dev/pts/2
[aprithepov@10 ~]$ ./semafor.sh 2 4 Выполнение > /dev/pts/1 &
[2] 3171
bash: /dev/pts/1: Отказано в доступе
[2]+  Выход 1          ./semafor.sh 2 4 Выполнение > /dev/pts/1
[aprithepov@10 ~]$ ./semafor.sh 2 4 Выполнение > /dev/pts/2 &
[2] 3176
bash: /dev/pts/2: Отказано в доступе
[2]+  Выход 1          ./semafor.sh 2 4 Выполнение > /dev/pts/2
[aprithepov@10 ~]$
```

изображение

```
[aprithepov@10 ~]$ ./semafor.sh 2 4 Ожидание
Следующее действие:
Ожидание
Следующее действие:
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие:
Выход
Выход
[aprithepov@10 ~]$
```

изображение

3. Используя встроенную переменную \$RANDOM, напомним командный файл, генерирующий случайную последовательность букв латинского алфавита. Для этого создадим файл random.sh и откроем его в emacs.

Напишем скрипт для выполнения 3 задания (рис.14).

```
#!/bin/bash
a=51
for ((i=0; i<$a; i++))
do
    ((char=$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
```

изображение

Проверим его работу (команда ./random.sh 158), предварительно дав ему право на выполнение с помощью команды chmod +x random.sh (рис.15).

```
[aprithepov@10 ~]$ chmod +x random.sh
[aprithepov@10 ~]$ ./random.sh 158
tbiefhlynpbpxixvedkpkppbohndgehzffkjigvenazaosuhojrjawysszemjmkjcrneqbpjukwia
bflmzpulpdytwejvqiayasyabthgmtximnurmcoejconzpnqookuvxmvzarvvpaafrwbwoprsp
[aprithepov@10 ~]$
```

изображение

Выводы

Я изучил основы программирования в оболочке ОС UNIX и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.