

Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Кузнецова С. В.

23 апреля 2023

Российский университет дружбы народов, Москва, Россия

Информация

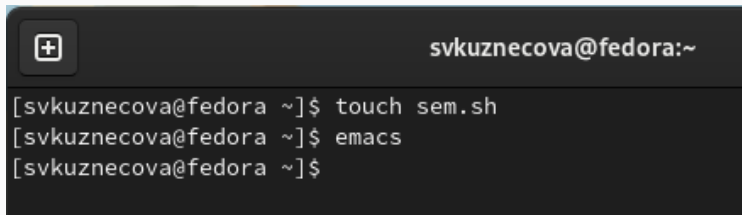
- Кузнецова София Вадимовна
- студент прикладной информатики
- Российский университет дружбы народов

Цель

Выполнение лабораторной работы

Создание первого файла для скрипта

Откроем терминал и создадим в домашнем каталоге файл `sem.sh`. После чего перейдём в `emacs`.

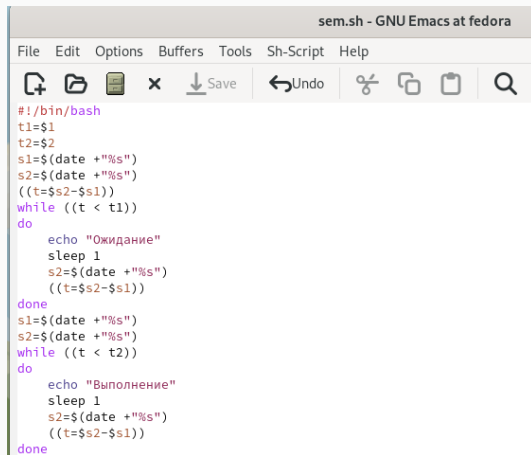
A terminal window with a dark background. The title bar shows a window icon with a plus sign and the text 'svkuznecova@fedora:~'. The terminal content shows three lines of commands and their prompts: '[svkuznecova@fedora ~]\$ touch sem.sh', '[svkuznecova@fedora ~]\$ emacs', and '[svkuznecova@fedora ~]\$'.

```
svkuznecova@fedora:~  
[svkuznecova@fedora ~]$ touch sem.sh  
[svkuznecova@fedora ~]$ emacs  
[svkuznecova@fedora ~]$
```

Рис. 1: Создание первого файла для скрипта

В emacs откроем созданный файл `sem.sh` и приступим к написанию командного файла, который реализует 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 привилегированном режиме.

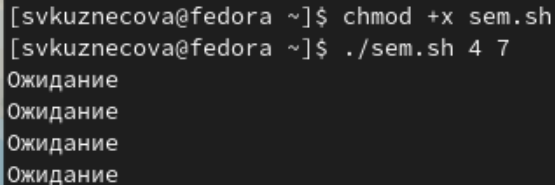
Написание первого скрипта



```
#!/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)
while ((t < t2))
do
    echo "Выполнение"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
```

Рис. 2: Написание первого скрипта

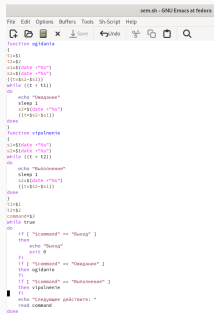
После того как скрипт написан мы сохраняем файл и запускаем emacs. В терминале мы даём этому файлу право на выполнение и запускаем его.



```
[svkuznecova@fedora ~]$ chmod +x sem.sh  
[svkuznecova@fedora ~]$ ./sem.sh 4 7  
Ожидание  
Ожидание  
Ожидание  
Ожидание
```

Рис. 3: Право на выполнение, запуск файла и проверка

Теперь нам нужно доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов. Снова открываем emacs и наш файл sem.sh.



```
sem.sh - GNU Emacs of Fedora
File Edit Options Buffers Tools Sh-Script Help
[Icons] Save Undo Copy Paste Find
function ogidante
{
  i=41
  i=42
  szk((date +%s))
  szk((date +%s))
  [[($i-$i)] ]
  while ((t < t1))
  do
    echo "Ожидание"
    sleep 1
    szk((date +%s))
    [[($i-$i)] ]
  done
}
function vipalente
{
  i=41
  i=42
  szk((date +%s))
  szk((date +%s))
  while ((t < t2))
  do
    echo "Восстановление"
    sleep 1
    szk((date +%s))
    [[($i-$i)] ]
  done
}
i=41
i=42
command=$1
while true
do
  if [ "$command" == "Bulud" ]
  then
    echo "Bulud"
    exit 0
  fi
  if [ "$command" == "Ожидание" ]
  then ogidante
  fi
  if [ "$command" == "Восстановление" ]
  then vipalente
  fi
  echo "Следующее действие: "
  read command
done
```

Рис. 4: Доработка первого скрипта

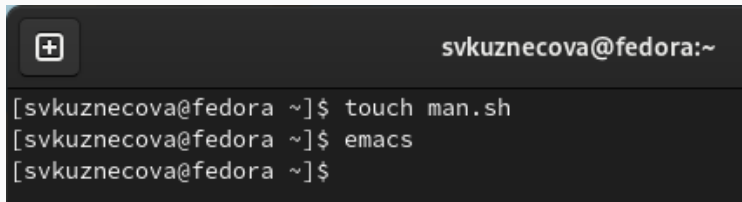
Сохраняем файл и проверяем его работу.

```
[ismahorin@fedora ~]$ emacs
[ismahorin@fedora ~]$ chmod +x sem.sh
[ismahorin@fedora ~]$ ./sem.sh 2 3 Ожидание > /dev/pts/1 &
[1] 132301
bash: /dev/pts/1: Отказано в доступе
[1]+ Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/1
[ismahorin@fedora ~]$ ./sem.sh 2 3 Ожидание > /dev/pts/2 &
[1] 132313
bash: /dev/pts/2: Отказано в доступе
[1]+ Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/2
[ismahorin@fedora ~]$ bash: /dev/pts/2: Отказано в доступе
bash: bash:: command not found...
similar command is: 'bash'
[ismahorin@fedora ~]$
```

Рис. 5: Сохранение и проверка

Создание файла для второго скрипта

В домашнем каталоге создаём файл `man.sh`, но уже для второго скрипта. Запускаем `emacs`.



```
svkuznecova@fedora:~  
[svkuznecova@fedora ~]$ touch man.sh  
[svkuznecova@fedora ~]$ emacs  
[svkuznecova@fedora ~]$
```

Рис. 6: Создание файла для второго скрипта

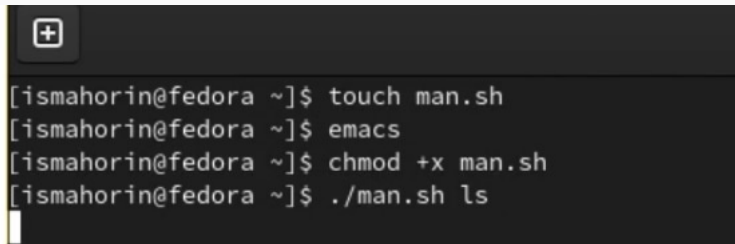
Приступаем к реализации команды `man` с помощью командного файла. Изучим содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое 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`.

fi" data-bbox="187 154 803 722"/>

```
man.sh - GNU Emacs at fedora  
File Edit Options Buffers Tools Sh-Script Help  
[Icons: Open, Save, Undo, etc.]  
#!/bin/bash  
c=$1  
if [ -f /usr/share/man/man1/$c.1.gz ]  
then  
    gunzip -c /usr/share/man/man1/$1.1.gz | less  
else  
    echo "Справки по данной команде нет"  
fi
```


Рис. 7: Написание второго скрипта

Сохраняем файл и даём в терминале право на выполнение. Запускаем файл `man.sh` для команды `ls`.

A terminal window with a dark background and a '+' icon in the top-left corner. It shows a series of commands and their outputs in a monospaced font.

```
[ismahorin@fedora ~]$ touch man.sh
[ismahorin@fedora ~]$ emacs
[ismahorin@fedora ~]$ chmod +x man.sh
[ismahorin@fedora ~]$ ./man.sh ls
```

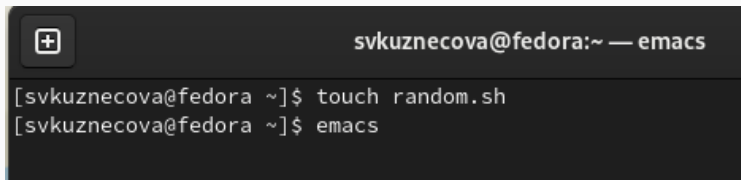
Рис. 8: Право на выполнение, запуск файла для команды `ls`



```
\\" DO NOT MODIFY THIS FILE!  It was generated by help2man 1.47.3.
TH LS "1" "March 2022" "GNU coreutils 8.32" "User Commands"
SH NAME
ls \- list directory contents
SH SYNOPSIS
B ls
[\fI\,OPTION\/\fR]... [\fI\,FILE\/\fR]...
SH DESCRIPTION
\" Add any additional description here
PP
list information about the FILES (the current directory by default).
sort entries alphabetically if none of \fB\-cftuvSUX\fR nor \fB\--sort\fR is specified.
PP
andatory arguments to long options are mandatory for short options too.
TP
\fB\-a\fR, \fB\--all\fR
do not ignore entries starting with .
TP
\fB\-A\fR, \fB\--almost-all\fR
do not list implied . and ..
TP
\fB\--author\fR
with \fB\-l\fR, print the author of each file
TP
```

Рис. 9: Информация о команде ls

Снова в домашнем каталоге создаём файл. Запускаем emacs.

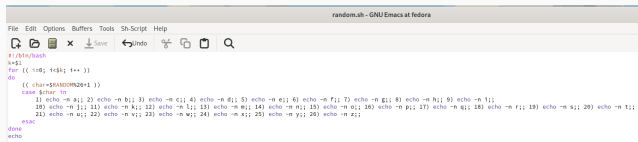


```
svkuznecova@fedora:~ — emacs  
[svkuznecova@fedora ~]$ touch random.sh  
[svkuznecova@fedora ~]$ emacs
```

Рис. 10: Создание третьего файла

Написание третьего скрипта

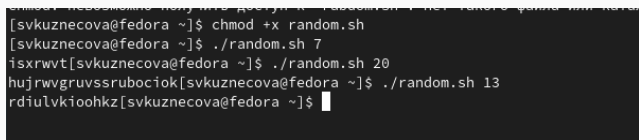
После открытия файла random.sh напишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтём, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.



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

Рис. 11: Написание третьего скрипта

Сохраняем наш скрипт и даём право на выполнение. Запускаем файл для трёх разных чисел.

A terminal window with a dark background and light-colored text. It shows a user named 'svkuznecova' on a 'fedora' machine. The user runs 'chmod +x random.sh' to make the script executable. Then they run './random.sh 7', './random.sh 20', and './random.sh 13'. Each command is preceded by a unique, long alphanumeric string. The output of the script is not visible.

```
[svkuznecova@fedora ~]$ chmod +x random.sh
[svkuznecova@fedora ~]$ ./random.sh 7
isxrwvt[svkuznecova@fedora ~]$ ./random.sh 20
hujrhwgruvssrubociok[svkuznecova@fedora ~]$ ./random.sh 13
rdiulvkioohkz[svkuznecova@fedora ~]$
```

Рис. 12: Право на выполнение, запуск файла

Вывод

Изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Спасибо за внимание!