

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №13

дисциплина: *Операционные системы*

Студент: ГАБРИЭЛЬ ТЬЕРРИ

Группа: НКНбд 01-20

МОСКВА 2021 г.

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. 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: semaphore.sh (Рисунки 1,2) и написал соответствующий скрипт.

```
tgabriel@tgabriel:~$ touch semaphore.sh
tgabriel@tgabriel:~$ emacs
```

```
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t1))
do
    echo "Pending"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t2 ))
do
    echo "Execution"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
```

Далее я проверил работу написанного скрипта (команда «./semaphore.sh 3 5»), предварительно добавив право на исполнение файла (команда «chmod +x semaphore.sh») (Рисунок 3)

```
tgabriel@tgabriel:~$ chmod +x semaphore.sh
tgabriel@tgabriel:~$ ./semaphore.sh 3 5
Pending
Pending
Pending
Execution
Execution
Execution
Execution
Execution
```

После этого я изменил скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверил его работу (например, команда «./semaphore.sh 2 3 Pending > /dev/pts/1 &») (Рисунки 4, 5, 6, 7)

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
function pending
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=$s2-$s1))
    while ((t < t1))
    do
        echo "Pending"
        sleep 1
        s2=$(date +%s)
        ((t=$s2-$s1))
    done
}
function execution
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=$s2-$s1))
    while ((t < t2 ))
    do
```

```
do
    echo "Execution"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Exit" ]
    then
        echo "Exit"
        exit 0
    fi
    if [ "$command" == "Pending" ]
    then pending
    fi
```

```

}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Exit" ]
    then
        echo "Exit"
        exit 0
    fi
    if [ "$command" == "Pending" ]
    then pending
    fi
    if [ "$command" == "Execution" ]
    then execution
    fi
    echo "following step: "
    read command
done

```

```

tgabriel@tgabriel:~$ emacs
tgabriel@tgabriel:~$ chmod +x semaphore.sh
tgabriel@tgabriel:~$ sudo ./semaphore.sh 2 3 Pending > /dev/pts/
1 &
[1] 15427
tgabriel@tgabriel:~$ -bash: /dev/pts/1: Permission denied

[1]+  Exit 1                  sudo ./semaphore.sh 2 3 Pending >
/dev/pts/1
tgabriel@tgabriel:~$ sudo ./semaphore.sh 3 4 Pending > /dev/pts/
2 &
[1] 15438
tgabriel@tgabriel:~$ sudo ./semaphore.sh 2 5 Execution > /dev/pt
s/2 &
[2] 15441
[1] Exit 1                  sudo ./semaphore.sh 3 4 Pending >
/dev/pts/2
tgabriel@tgabriel:~$ -bash: /dev/pts/2: Permission denied
tgabriel@tgabriel:~$

```

2. Реализовал команду man с помощью командного файла. Изучил содержимое каталога /usr/share/man/man1 (Рисунки 8,9). 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.

```
tgabriel@tgabriel:~$ cd /usr/share/man
tgabriel@tgabriel:/usr/share/man$ ls
cs  fi          hu  ko      man4  man8  pt_BR  sr  zh_CN
da  fr          id  man1  man5  nl    ro    sv  zh_TW
de  fr.ISO8859-1 it  man2  man6  pl    ru    tr
es  fr.UTF-8    ja  man3  man7  pt    sl    uk
tgabriel@tgabriel:/usr/share/man$ cd man1
tgabriel@tgabriel:/usr/share/man/man1$ ls
'[,1.gz'
aa-enabled.1.gz
aa-exec.1.gz
aa-features-abl.1.gz
aconnect.1.gz
add-apt-repository.1.gz
addr2line.1.gz
airscan-discover.1.gz
alsabat.1.gz
alsactl.1.gz
alsaloop.1.gz
```

```
alsaloop.1.gz
alsamixer.1.gz
alsatplg.1.gz
alsaucm.1.gz
amidl.1.gz
amixer.1.gz
apg.1.gz
apgbfm.1.gz
aplay.1.gz
aplaymidi.1.gz
apport-bug.1.gz
apport-cli.1.gz
apport-collect.1.gz
apport-unpack.1.gz
appres.1.gz
appstreamcli.1.gz
apropos.1.gz
apt-add-repository.1.gz
apt-d.1.gz
apt-dcon.1.gz
apt-extracttemplates.1.gz
apt-ftparchive.1.gz
```

Для данной задачи я создал файл: man.sh (Рисунки 10) и написал соответствующий скрипт.(Рисунок 11)

```
tgabriel@tgabriel:~$ touch man.sh
tgabriel@tgabriel:~$ emacs
```

```
#!/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 "There is no help for this command"
fi
```

Далее я проверил работу написанного скрипта (команды «./man.sh ls» и «./man.sh mkdir»), предварительно добавив право на исполнение файла (команда «chmod +x man.sh») (Рисунки 12, 13, 14)

```
tgabriel@tgabriel:~$ chmod +x man.sh
tgabriel@tgabriel:~$ ./man.sh ls
tgabriel@tgabriel:~$ ./man.sh mkdir
tgabriel@tgabriel:~$
```

```
.\ " DO NOT MODIFY THIS FILE!  It was generated by help2man 1.47.
3.
.TH LS "1" "December 2020" "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
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\-\a\|fR, \fB\-\-all\|fR
do not ignore entries starting with .
:
```



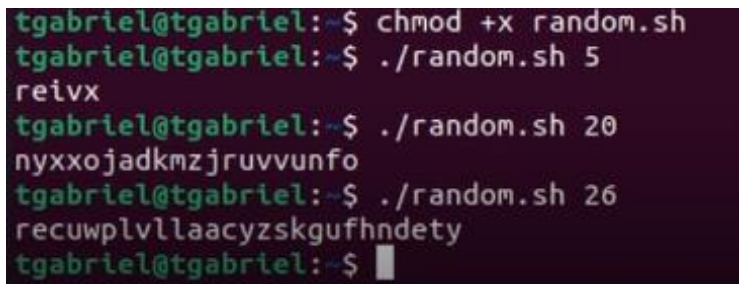
```
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.
3.
.TH MKDIR "1" "December 2020" "GNU coreutils 8.32" "User Command
s"
.SH NAME
mkdir \- make directories
.SH SYNOPSIS
.B mkdir
[\fI\,OPTION\[/\fR]... \fI\,DIRECTORY\[/\fR...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Create the DIRECTORY(ies), if they do not already exist.
.PP
Mandatory arguments to long options are mandatory for short opti
ons too.
.TP
\fB\m\fR, \fB\-\mode\fR=\fI\,MODE\[/\fR
set file mode (as in chmod), not a=rwx \- umask
.TP
\fB\p\fR, \fB\-\parents\fR
:
```

3. Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создал файл: random.sh (Рисунок 15) и написал соответствующий скрипт (Рисунок 16)

```
tgabriel@tgabriel:~$ touch random.sh
tgabriel@tgabriel:~$ emacs
```

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
k=$1
for (( i=0; i<$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
```

Далее я проверил работу написанного скрипта (команды «./random.sh 5» «./random.sh 20» и «./random.sh 26») предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (Рисунок 17)



```
tgabriel@tgabriel:~$ chmod +x random.sh
tgabriel@tgabriel:~$ ./random.sh 5
reivx
tgabriel@tgabriel:~$ ./random.sh 20
nyxhojadkmzjrurvunfo
tgabriel@tgabriel:~$ ./random.sh 26
recuwplvllaacyzskgufhndety
tgabriel@tgabriel:~$
```

Вывод: > В ходе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX, а также научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

Контрольные вопросы:

1. while [\$1 != "exit"] В данной строчке допущены следующие ошибки:
 - не хватает пробелов после первой скобки [и перед второй скобкой]
 - выражение \$1 необходимо взять в "", потому что эта переменная может содержать пробелы Таким образом, правильный вариант должен выглядеть так: while ["\$1" != "exit"]
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:
 - Первый: VAR1="Hello," VAR2=" World" VAR3="VAR1VAR2" echo "\$VAR3"
Результат: Hello, World
 - Второй: VAR1="Hello," VAR1+=" World" echo "\$VAR1" Результат: Hello, World
3. Команда seq в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры:
 - seq LAST: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение is не выдает.
 - seq FIRST LAST: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных.
 - seq FIRST INCREMENT LAST: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT . Если LAST меньше, чем FIRST, он не производит вывод.
 - seq -f «FORMAT» FIRST INCREMENT LAST: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными.

- `seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО`: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными.
 - `seq -w FIRST INCREMENT LAST`: эта команда используется для выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.
4. Результатом данного выражения `$((10/3))` будет 3, потому что это целочисленное деление без остатка.
 5. Отличия командной оболочки `zsh` от `bash`:
 - В `zsh` более быстрое автодополнение для `cd` с помощью `Tab`
 - В `zsh` существует калькулятор `zcalc`, способный выполнять вычисления внутри терминала
 - В `zsh` поддерживаются числа с плавающей запятой
 - В `zsh` поддерживаются структуры данных «хэш»
 - В `zsh` поддерживается раскрытие полного пути на основе неполных данных
 - В `zsh` поддерживается замена части пути
 - В `zsh` есть возможность отображать разделенный экран, такой же как разделенный экран `vim`
 6. `for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать `$` перед переменными `()`.
 7. Преимущества скриптового языка `bash`:
 - Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS
 - Удобное перенаправление ввода/вывода
 - Большое количество команд для работы с файловыми системами Linux
 - Можно писать собственные скрипты, упрощающие работу в Linux
 Недостатки скриптового языка `bash`:
 - Дополнительные библиотеки других языков позволяют выполнить больше действий
 - Bash не является языком общего назначения
 - Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта
 - Скрипты, написанные на `bash`, нельзя запустить на других операционных системах без дополнительных действий