

Программирование в командном процессоре ОС UNIX. Расширенное программирование

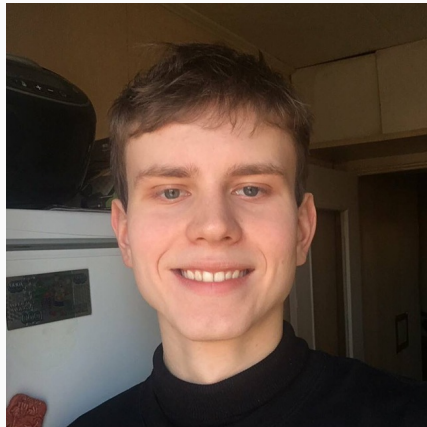
Лабораторная работа №12

Татур С. А.

Российский университет дружбы народов, Москва, Россия

Информация

- Татур Стефан Андреевич
- студент 1 курса, группа НММбд-03-22
- Российский университет дружбы народов



Вводная часть

- Командный процессор ОС UNIX
- Командные файлы

- Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

- Ознакомиться с теоретическим материалом.
- Выполнить упражнения.
- Ответить на контрольные вопросы.

Выполнение лабораторной работы №12

Первая программа

```
~]$ touch lab12_1.sh
```

```
Открыть ▾ + lab12_1.sh
~/
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t1)) do
    echo "Ожидайте"
    sleep 1
    s2=$(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
```

```
Ожидайте
Ожидайте
Ожидайте
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
```


Первая программа доработка

```
Открыть + lab12_1.sh ~/
1 #!/bin/bash
2 t1=$1
3 t2=$2
4 s1=$(date +%s")
5 s2=$(date +%s")
6 ((t=s2-s1))
7 while ((t < t1)) do
8     echo "Ожидайте"
9     sleep 1
10    s2=$(date +%s")
11    ((t=s2-s1))
12 done
13 s1=$(date +%s")
14 s2=$(date +%s")
15 ((t=s2-s1))
16 while ((t < t2)) do
17     echo "Выполнение"
18     sleep 1
19    s2=$(date +%s")
20    ((t=s2-s1))
21 done
22 t1=$1
23 t2=$2
24 command=$3
25 while true
26 do
27     if [ "$command" = "Выход" ]
28     then echo "Выход"
29         exit 0
30     fi
31     if [ "$command" = "Ожидание" ]
32     then ogidanie
33     fi
34     if [ "$command" = "Выполнение" ]
35     then vipolnenie
36     fi
37     echo "Следующее действие"
38     read command
39 done
```

```
:.1.gz
'.1.gz'
ab.1.gz
abrt.1.gz
abrt-action-analyze-backtrace.1.gz
abrt-action-analyze-c.1.gz
abrt-action-analyze-ccpp-local.1.gz
abrt-action-analyze-core.1.gz
abrt-action-analyze-java.1.gz
abrt-action-analyze-oops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-vmcore.1.gz
abrt-action-analyze-vulnerability.1.gz
abrt-action-analyze-xorg.1.gz
abrt-action-check-oops-for-hw-error.1.gz
abrt-action-find-bodhi-update.1.gz
abrt-action-generate-backtrace.1.gz
abrt-action-generate-core-backtrace.1.gz
abrt-action-install-debuginfo.1.gz
abrt-action-list-dsos.1.gz
```

Вторая программа

```
~]$ touch lab12_2.sh
```

```
Открыть ▾  lab12_2.sh  
#!/bin/bash  
c=$1  
if [ -f /usr/share/man/man1/${c}.1.gz ]  
then  
    gunzip -c /usr/share/man/man1/${c}.1.gz | less  
else  
    echo "Справки по данной команде нет"  
fi
```


Вторая программа

```
~]$ chmod +x lab12_2.sh
```

```
[10]+  Остановлен ./lab12_2.sh ls
```

```
/* DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
TH LS "1" "January 2023" "GNU coreutils 9.0" "User Commands"
.SH NAME
ls - list directory contents
.SH SYNOPSIS
ls
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 \[FB\]-ctuvSUX\[FR\] nor \[FB\]-\[sort\]\[FR\] is specified.
Mandatory 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
\[FB\]-b\[FR\], \[FB\]-\[escape\]\[FR\]
print C-style escapes for nongraphic characters
```

Третья программа

```
Открыть ▾  lab12_3.sh
#!/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;;
    esac
done
```

```
$ chmod +x lab12_3.sh
```

```
lab12_3.sh 10
bqauyjp
```

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 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`

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` 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

Результаты

В ходе выполнения лабораторной работы были изучены основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.