

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Лабораторная работа №12

Старцева А. С.

29 апреля 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Старцева Алина Сергеевна
- 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

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
```

```
[astarceva@asstarceva ~]$ chmod +x lab12_1.sh
```

```
[astarceva@asstarceva ~]$ ./lab12_1.sh 5 3
```

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

Первая программа доработка

```
Открыть ▾  • lab12_1.sh  
~/  
    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 [ "$command" == "Ожидание" ]  
    then pass  
    fi  
    if [ "$command" == "Выполнение" ]  
    then pass  
    fi  
    echo "Следующее действие"  
    read command  
done  
.
```

```
[astarceva@asstarceva ~]$ ./lab12_1.sh 5 3 4  
Ожидайте  
Ожидайте  
Ожидайте  
Ожидайте  
Выполнение  
Выполнение  
Следующее действие  
  
Следующее действие  
  
Следующее действие  
  
Следующее действие
```

Вторая программа

```
[astarceva@astarceva ~]$ cd /usr/share/man/man1
[astarceva@astarceva man1]$ ls
i-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-cpp-locale.1.gz
abrt-action-analyze-core.1.gz
abrt-action-analyze-java.1.gz
abrt-action-analyze-ops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-uncore.1.gz
abrt-action-analyze-vulnerability.1.gz
abrt-action-analyze-xorg.1.gz
abrt-action-check-ops-for-hw-error.1.gz
abrt-action-find-budhi-update.1.gz
abrt-action-generate-backtrace.1.gz
abrt-action-generate-core-backtrace.1.gz
abrt-action-install-debuginfo.1.gz
```

```
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

```
astarceva@asstarceva ~]$ chmod +x lab12_2.sh
```

```
astarceva@asstarceva ~]$ ./lab12_2.sh ls
astarceva@asstarceva ~]$ ./lab12_2.sh cd
```

```
astarceva@asstarceva:~/bin/bash ./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
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 -l -sort fr is s
pecified.
PP
Mandatory arguments to long options are mandatory for short options too.
TP
fb -a fr, fb -l -all fr
do not ignore entries starting with .
TP
fb -A fr, fb -l -almost -all fr
do not list implied . and ..
TP
fb -l -author fr
```

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;;
```

```
[astarceva@asstarceva ~]$ chmod +x lab12_3.sh
```

```
[astarceva@asstarceva ~]$ ./lab12_3.sh 8
bf1dxqrp
[astarceva@asstarceva ~]$ ./lab12_3.sh 3
chc
```

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`

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="${VAR1} ${VAR2}" echo "$VAR3"` :
Hello, World : `VAR1 = "Hello", "VAR1+= "World" echo "$VAR1"` Результат: Hello, World

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.