Лабораторная работа №12

Покрас Илья Михайлович

Цель работы

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

Последовательность работы

```
#!/bin/bash
lockfile="./lockfile"
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do
    echo "no lcok"
    sleep 1
    flock -n ${fn}
done
for ((i=0; i<5; i++))
do
    echo "work"
    sleep 1
done
```

Script 1

Я написал командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом. Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой, в котором также запущен этот файл, но не фоновом, а в привилегированном режиме.

```
impokras@impokras-VirtualBox:~/lab13$ ./script1.sh
lock
work
work
work
work
work
impokras@impokras-VirtualBox:~/lab13$
```

Script 1 result

```
#!/bin/bash
cd /usr/share/man/man1
less $1*
```

Script 2

Я реализовал команду man с помощью командного файла. Изучил 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

```
test \- check file types and compare values
.SH SYNOPSIS
.B test
.I EXPRESSION
.br
.B test
.br
.\" \& tells doclifter the brackets are literal (Bug#31803).
.B [\&
.I EXPRESSION
.B ]\&
.br
.B "[\& ]\&"
.br
.B [\&
.I OPTION
.SH DESCRIPTION
.\" Add any additional description here
.PP
Exit with the status determined by EXPRESSION.
.TP
\fB\-\-help\fR
display this help and exit
.TP
\fB\-\-version\fR
output version information and exit
.PP
An omitted EXPRESSION defaults to false.  Otherwise,
EXPRESSION is true or false and sets exit status.  It is one of:
.TP
( EXPRESSION )
EXPRESSION is true
.TP
! EXPRESSION
EXPRESSION is false
.TP
EXPRESSION1 \fB\-a\fR EXPRESSION2
both EXPRESSION1 and EXPRESSION2 are true
.TP
EXPRESSION1 \fB\-o\fR EXPRESSION2
either EXPRESSION1 or EXPRESSION2 is true
.TP
\fB\-n\fR STRING
the length of STRING is nonzero
.TP
```

Script 2 result

```
#!/bin/bash
M=10
c=1
d=1
echo
echo "10 random words:"
while (($c!=(($M+1))))
      do
          echo $(for((i=1;i<=10;i++)) do printf '%s' "${RANDOM:0:1}"; done) | tr '[0-9]' '[a-z]'
       echo $d
       ((c+=1))
       ((d+=1))
      done
```

Script 3

Используя встроенную переменную $RANDOM, я написал командный файл, генерирующий случайную последовательность букв латинского алфавита.

```
impokras@impokras-VirtualBox:~/lab13$ ./script3.sh

10 random words:
cibccbgedc
1
cbfbcffcbb
2
gddhbcfbcc
3
bjcbbddcgc
4
cbbcdcdccb
5
dbedjjccbc
6
ccbcfbccdb
7
bhcigifbbc
8
chicbgcbci
```

Script 3 result

Вывод

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

Контрольные вопросы:

1.Нужны пробелы после и перед открывающей и закрывающей скобками соответственно. Также желательно заключить $1 в кавычки (“$1”) во избежание ошибки, если 1пуст. 2. $Str0 = $”str1$str2” 3.Seq выводит последовательность целых чисел с шагом, заданным пользователем. Другая утилита с той же функцией – jot. 4.Результатом будет 3. 5.В zsh можно настраивать горячие клавиши. Автодополнение более сложное и гибкое. Используется большое количество различных опций, а также максимально краткий синтаксис. В итоге, zsh удобен для повседневной, рутинной работы, а для написания скриптов всё же лучше использовать bash. 6.Синтаксис верен. 7.Bash имеет достаточно много сходств другими языками. Недостатком является достаточно нагруженный синтаксис (легко допустить ошибку, потеряв, допустим, fi). В целом он достаточно понятен, однако он выглядит не совсем обычно в некоторых моментах, также необходимо читать много справок, чтобы на нём писать.