Отчет по лабораторной работе №13

Тема:

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: Операционные системы

Студент:Яссин Оулед Салем

Группа: НПИБД02-20

Москва, 2021г.

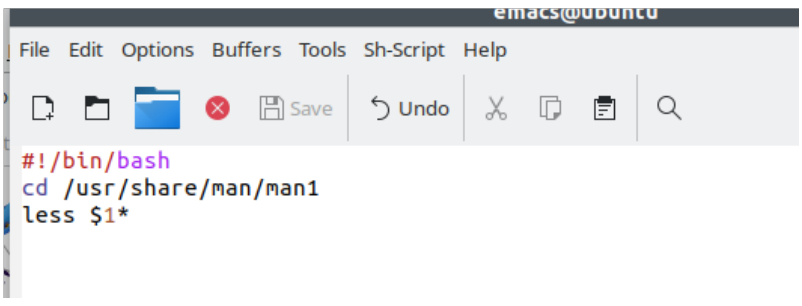
Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов Ход работы 1. Упрощённый механизм подобный семафору для доступа к некоторому ресурсу

```
#!/bin/bash
x="./x"
exec {fn}>$x
echo "block"
until flock -n ${fn}
do
    echo "not blocked "
    sleep 1
    flock -n ${fn}
done
for ((i = 0; i <= 5; i++))
do
    echo "work"
    sleep 1
done
```

```
yassine@ubuntu:~$ bash lab013.sh
block
work
work
work
work
work
work
yassine@ubuntu:~$
```

1. man с помощью командного файла



```
yassine@ubuntu:~$ emacs
yassine@ubuntu:~$ touch lab13_2.sh
yassine@ubuntu:~$ chmod +x lab13_2.sh
yassine@ubuntu:~$ emacs
yassine@ubuntu:~$ bash lab013_2.sh
```

```
.\" DO NOT MODIFY THIS FILE!  It was generated by help2man 1.47.3.
.TH LS "1" "September 2019" "GNU coreutils 8.30" "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 sp
ecified.
.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 .
.TP
\fB\-A\fR, \fB\-\-almost\-all\fR
do not list implied . and ..
.TP
\fB\-\-author\fR
:
```

1. Командный фаил , генерирующий строку длиной 20 со случайной последовательностью букв латинского алфавита(STR_LEN первоначально был задан $RANDOM , но создает слишком длинные строки)

```
File  Edit  Options  Buffers  Tools  Sh-Script  Help
Save  Undo
#!/bin/bash
m= 10
a= 1
b= 1
echo "10 рандомных слов: "
while (($a!=($m + 1)))
do
echo $(for((i= 1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done) | tr '[0-9]'
 '[a-z]'
echo $b
    ((a+=1))
    ((b+=1))
    done
```

```
yassine@ubuntu:~$ emacs
yassine@ubuntu:~$ bash lab13_3.sh
10 рандомных слов:
bcbfbbdchb
1
bccbdgbfic
2
bcbccbbhce
3
bdceccdecb
4
cbcbjdcbcc
5
bcbbcbcbbb
6
bbcbbcfdcc
7
eccfbbcdbb
8
cbbccbbcdb
9
dbgfdcccbb
10
yassine@ubuntu:~$
```

Вывод

В результате работы, я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов Контрольные вопросы 1. $1 следует внести в кавычки. 2. С помощью знака >,| можно объединить несколько строк в одну. 3. Эта утилита выводит последовательность целых чисел с заданным шагом. Также можно реализовать с помощью утилиты jot. 4. Результатом вычисления выражения $((10/3)) будет число 3. 5. В zsh можно настроить отдельные сочетания клавиш так, как вам нравится. Использование истории команд в zsh ничем особенным не отличается от bash. Zsh очень удобен для повседневной работы и делает добрую половину рутины за вас. Но стоит обратить внимание на различия между этими двумя оболочками. Например, в zsh после for обязательно вставлять пробел, нумерация массивов в zsh начинается с 1, чего совершенно невозможно понять. Так, если вы используете shell для повседневной работы, исключающей написание скриптов, используйте zsh. Если вам часто приходится писать свои скрипты, только bash! Впрочем, можно комбинировать. 6. Синтаксис конструкции for ((a=1; a <= LIMIT; a++)) верен. 7. Язык bash и другие языки программирования: a. Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на си/си++, скомпилированных с максимальной оптимизацией; b. Скорость работы виртуальной ява-машины с байт-кодом часто превосходит скорость аппаратуры с кодами, получаемыми трансляторами с языков высокого уровня. Ява-машина уступает по скорости только ассемблеру и лучшим оптимизирующим трансляторам; c. Скорость компиляции и исполнения программ на яваскрипт в популярных браузерах лишь в 2-3 раза уступает лучшим трансляторам и превосходит даже некоторые качественные компиляторы, безусловно намного (более чем в 10 раз) обгоняя большинство трансляторов других языков сценариев и подобных им по скорости исполнения программ; d. Скорость кодов, генерируемых компилятором языка си фирмы Intel, оказалась заметно меньшей, чем компилятора GNU и иногда LLVM; e. Скорость ассемблерных кодов x86-64 может меньше, чем аналогичных кодов x86, примерно на 10%; f. Оптимизация кодов лучше работает на процессоре Intel; g. Скорость исполнения на процессоре Intel была почти всегда выше, за исключением языков лисп, эрланг, аук (gawk, mawk) и бэш. Разница в скорости по бэш скорее всего вызвана разными настройками окружения на тестируемых системах, а не собственно транслятором или железом. Преимущество Intel особенно заметно на 32-разрядных кодах; h. Стек большинства тестируемых языков, в частности, ява и яваскрипт, поддерживают только очень ограниченное число рекурсивных вызовов. Некоторые трансляторы (gcc, icc, ...) позволяют увеличить размер стека изменением переменных среды исполнения или параметром; i. В рассматриваемых версиях gawk, php, perl, bash реализован динамический стек, позволяющий использовать всю память компьютера. Но perl и, особенно, bash используют стек настолько экстенсивно, что 8-16 ГБ не хватает для расчета ack(5,2,3)