

Лабораторная работа №13

Журавлев Георгий Иванович

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.

FLOCK(1)	User Commands	FLOCK(1)
NAME		
flock - manage locks from shell scripts		
SYNOPSIS		
flock [options] file directory command [arguments]		
flock [options] file directory -c command		
flock [options] number		
DESCRIPTION		
This utility manages flock(2) locks from within shell scripts or from the command line.		
The first and second of the above forms wrap the lock around the execution of a command, in a manner similar to su(1) or newgrp(1). They lock a specified file or directory, which is created (assuming appropriate permissions) if it does not already exist. By default, if the lock cannot be immediately acquired, flock waits until the lock is available.		
The third form uses an open file by its file descriptor number. See the examples below for how that can be used.		
OPTIONS		
-c, --command command		
Pass a single command, without arguments, to the shell with -c.		
-E, --conflict-exit-code number		
The exit code used when the -n option is in use, and the conflicting lock exists, or the -w option is in use, and the timeout is reached. The default value is 1.		
-F, --no-fork		
Do not fork before executing command. Upon execution the flock process is replaced by command which continues to hold the lock. This option is incompatible with --close as there would otherwise be nothing left to hold the lock.		
-e, -x, --exclusive		
Obtain an exclusive lock, sometimes called a write lock. This is the default.		
-n, --nb, --nonblock		
Fail rather than wait if the lock cannot be immediately acquired. See the -E option for the exit code used.		
SLEEP(1)	User Commands	SLEEP(1)
NAME		
sleep - delay for a specified amount of time		
SYNOPSIS		
sleep NUMBER[SUFFIX]...		
sleep OPTION		
DESCRIPTION		
Pause for NUMBER seconds. SUFFIX may be 's' for seconds (the default), 'm' for minutes, 'h' for hours or 'd' for days. Unlike most implementations that require NUMBER be an integer, here NUMBER may be an arbitrary floating point number. Given two or more arguments, pause for the amount of time specified by the sum of their values.		
--help display this help and exit		
--version		
output version information and exit		
AUTHOR		
Written by Jim Meyering and Paul Eggert.		
REPORTING BUGS		
GNU coreutils online help: < https://www.gnu.org/software/coreutils/ >		
Report sleep translation bugs to < https://translationproject.org/team/ >		
COPYRIGHT		
Copyright © 2018 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later < https://gnu.org/licenses/gpl.html >.		
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.		
SEE ALSO		
sleep(3)		

```
#!/bin/bash
lockfile="./lockfile"
fn=
echo name of the file:
read fn
exec ${fn} > $lockfile
echo locked
until flock -n ${fn}
do
    echo not locked
    sleep 0.05m
    flock -n ${fn}
done
t1=
echo time:
read t1
for(( i=0; i<=t1; i++))
do
    echo working
    sleep 0.05m
done
```

U:--- lab13.01.sh All L6 (Shell-script[sh])

2. Реализовал команду `man` с помощью командного файла. Изучил содержимое каталога `/usr/share/man/man1`.

```
#!/bin/bash
cd /usr/share/man/man1
command=""
echo command that u need:
less $command*
```

3. Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

```
#!/bin/bash
Words=
echo number of random words:
read Words
for((i=0;i<$Words;i++))
do echo $RANDOM | tr '[0-32670]' '[a-z]'
done
```

-- lab13.03.sh All L8 (Shell-script[bash])

5

```
g_zhuravlev@g ➔ ./lab13.03.sh 961 14:54:1
number of random words:
6
b84gd
bed58
b9f8
ge9g
eh9f
8f44
g_zhuravlev@g ➔ 966 15:03:3
```

Вывод.

Благодаря этой лабораторной работе, я написал некоторые интересные скрипты, которые оказались сложнее предыдущих; развился в сфере взаимодействия с bash.

Контрольные вопросы.

1. В строке `while [$1 != "exit"]` квадратные скобки надо заменить на круглые.
2. пример,

```
str1="Goodbye, "  
str2="Moon"  
str3="$str1$str2"  
echo "$str3"
```
3. Команда `seq` выводит последовательность целых или действительных чисел, подходящую для передачи
4. $\$((10/3)) = 3$;
5. `zsh` VS `bash`
 1. ZSH
 - 5.1. `zmv` - поможет массово переименовать файлы/директории.
 - 5.2. `zcalc` — это замечательный калькулятор командной строки, удобный способ считать быстро, не покидая
 - 5.3. `Autopushd` позволяет делать `popd` после того, как с помощью `cd`, чтобы вернуться в предыдущую дир
 - 5.4. Поддержка для структур данных «хэш».
 - 5.5. Поддержка чисел с плавающей точкой.
 2. `bash`
 - 5.6. Использование опции `-rcfile <filename>` с `bash` позволяет исполнять команды из определённого файла
 - 5.7. Может быть вызвана командой `sh`.
 - 5.8. Можно запустить в определённом режиме POSIX.
 - 5.9. Можно включить в режиме ограниченной оболочки (с `rbash` или `--restricted`).
 - 5.10. Перенаправление вывода с использованием операторов `>`, `>|`, `<>`, `>&`, `&>`, `>>`.
6. Синтаксис конструкции `-` верен.
 - 7.1. Стек большинства тестируемых языков поддерживают только очень ограниченное число рекурсивных
 - 7.2. В `bash` реализован динамический стек, позволяющий использовать всю память компьютера.
 - 7.3. Скорость `bash` кодов x86-64 может меньше, чем аналогичных кодов x86.
 - 7.4. Скорость компиляции и исполнения программ на яваскрипт в популярных браузерах лишь в 2-3 раза уступает лучшим трансляторам и превосходит даже некоторые качественные компиляторы, более чем