

Лабораторная работа № 13

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Маметкадыров Ынтымак

Цель работы

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

Выполнение лабораторной работы

1. Написали командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t_1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (рис. 1). Затем проверили работу командного файла (рис. 2).

```

t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t1))
do
    echo "Ожидание"
    sleep 1
    s2=$(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

```

Рис. 1. Листинг командного файла `semaf.sh`

```

[itmametkadihrov@itmametkadihrov ~]$ ./semaf.sh 3 4
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
[itmametkadihrov@itmametkadihrov ~]$

```

Рис. 2. Работа командного файла `semaf.sh`

2. Доработали командный файл так, чтобы ее можно было выполнять в нескольких терминалах (рис. 3). Запустили командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой.

```
#!/bin/bash
function ojidanie
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t < t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
function vpolnenie
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t < t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(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 [ "$command" == "Ожидание" ]
then ojidanie
fi
if [ "$command" == "Выполнение" ]
then vpolnenie
fi
echo "Следующее действие: "
read command
done

```

Рис. 3. Командный файл *semaf.sh*

3. Реализовали команду *man* с помощью командного файла (рис. 4). Командный файл получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде, например, если ввести команду *./man.sh ls* (рис. 6), то выведется справка о команде *ls*, (рис. 5) или сообщение об отсутствии справки, если соответствующего файла нет в каталоге *man1* (рис. 6).

```

if ( test -f /usr/share/man/man1/$1.1.gz )
then gunzip -c /usr/share/man/man1/$1.1.gz | less
else
echo "Справка по данной команде отсутствует"
fi

```

Рис. 4. Командный файл *man.sh*

```
Файл Правка Вид Поиск Терминал Справка
<hare/man/man1/ls.1.gz" [только для чтения][noeol] 19L, 3213C
Обнаружена ошибка при обработке function gzip#read:
строка 51:
E325: ВНИМАНИЕ
Обнаружен своп-файл с именем "/var/tmp/ls.1.gz.swp"
    владелец: itmametskadihrov    дата: Fri Jun  4 22:13:17 2021
    имя файла: /usr/share/man/man1/ls.1.gz
    изменён: ДА
    пользователь: itmametskadihrov компьютер: itmametskadihrov
    процесс: 15982 (ещё выполняется)
при открытии файла: "/usr/share/man/man1/ls.1.gz"
    дата: Tue Nov 17 01:24:47 2020

(1) Возможно, редактирование этого же файла выполняется в другой программе.
    Если это так, то будьте внимательны при внесении изменений, чтобы
    у вас не появилось два разных варианта одного и того же файла.
    Завершите работу или продолжайте с осторожностью.
(2) Сеанс редактирования этого файла завершён аварийно.
    В этом случае, используйте команду ":recover" или "vim -r /usr/share/man/man
1/ls.1.gz"
    для восстановления изменений (см. ":help recovery").
    Если вы уже выполняли эту операцию, удалите своп-файл "/var/tmp/ls.1.gz.swp"

-- Продолжение следует --
```

Рис. 5. Результат команды `./man.sh ls`

```
[itmametskadihrov@itmametskadihrov ~]$ ./man.sh ls

[5]+  Stopped                  ./man.sh ls
[itmametskadihrov@itmametskadihrov ~]$ ./man.sh lsкуцак
Справка по данной команде отсутствует
[itmametskadihrov@itmametskadihrov ~]$
```

Рис. 6. Сообщение об отсутствии справки

- Используя встроенную переменную `$RANDOM`, написали командный файл (рис. 7), генерирующий случайную последовательность букв латинского алфавита (рис. 8).

```
random.sh - emacs@itmametskadihrov
File Edit Options Buffers Tools Sh-Script Help
[Icons: File Explorer, Save, Undo, Cut, Copy, Paste, Find]
head /dev/random | tr -dc A-Za-z | head -c 13 ; echo ''
```

Рис. 7. Командный файл `random.sh`

```
[itmametskadihrov@itmametskadihrov ~]$ ./random.sh
EPudrOVrUaERB
[6] Done emacs random.sh
[itmametskadihrov@itmametskadihrov ~]$ ./random.sh
kiHiNjklLLsfs
[itmametskadihrov@itmametskadihrov ~]$ ./random.sh
omghBUSGurKsN
[itmametskadihrov@itmametskadihrov ~]$ ./random.sh
KBoNVWaENblUM
[itmametskadihrov@itmametskadihrov ~]$ █
```

Рис. 8. Результат работы командного файла random.sh

Вывод

Изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Ответы на контрольные вопросы

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="\$VAR1VAR2" echo "\$VAR3"

Результат: Hello, World

Второй: VAR1="Hello," VAR1+=" World" echo "\$VAR1"

Результат: Hello, World.

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 ВКЛЮЧЕНО: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными.
 - seq -w FIRST INCREMENT LAST: эта команда используется для выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.
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 с 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 ().
7. Преимущества скриптового языка bash:

- Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS
- Удобное перенаправление ввода/вывода
- Большое количество команд для работы с файловыми системами Linux
- Можно писать собственные скрипты, упрощающие работу в Linux

Недостатки скриптового языка bash:

- Дополнительные библиотеки других языков позволяют выполнить больше действий
- Bash не является языком общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта
- Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий