

Отчёт по лабораторной работе №14

Петлин Артём Дмитриевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	semaphor.sh	9
4.2	работа скрипта	10
4.3	работа скрипта	10
4.4	new semaphor.sh	11
4.5	new semaphor.sh	12
4.6	работа скрипта	12
4.7	работа скрипта	12
4.8	работа скрипта	13
4.9	reman.sh	13
4.10	работа скрипта	14
4.11	randoml.sh	14
4.12	работа скрипта	15

Список таблиц

1 Цель работы

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

2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите 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`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до

32767.

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

4 Выполнение лабораторной работы

```
#!/bin/bash
LOCK_FILE="/tmp/resource.lock"
T1=5
T2=3

echo "Процесс $$ ожидает освобождения ресурса..."
while [ -f "$LOCK_FILE" ] && [ $T1 -gt 0 ]; do
    echo "Ресурс занят, ожидание..."
    sleep 1
    ((T1--))
done

if [ $T1 -le 0 ]; then
    echo "Процесс $$: время ожидания истекло, выход"
    exit 1
fi

touch "$LOCK_FILE"
echo "Процесс $$ получил доступ к ресурсу"

echo "Процесс $$ использует ресурс..."
sleep $T2

rm -f "$LOCK_FILE"
echo "Процесс $$ освободил ресурс"
```

Рис. 4.1: semaphor.sh

Пишем командный файл, реализующий упрощённый механизм семафоров.

```
[adpetlin@adpetlin ~]$ chmod +x semaphor.sh
[adpetlin@adpetlin ~]$ ./semaphor.sh > /dev/tty2
[adpetlin@adpetlin ~]$ |
```

Рис. 4.2: работа скрипта

Перенаправляем вывод из первого терминала во второй.

```
[adpetlin@adpetlin ~]$ ./semaphor.sh
Процесс 10438 ожидает освобождения ресурса...
Процесс 10438 получил доступ к ресурсу
Процесс 10438 использует ресурс...
Процесс 10438 освободил ресурс
[adpetlin@adpetlin ~]$ |
```

Рис. 4.3: работа скрипта

Вывод во втором терминале.

```
#!/bin/bash

LOCK_FILE="/tmp/resource.lock"
FIFO_DIR="/tmp/resource_fifos"
T1=5
T2=3

mkdir -p "$FIFO_DIR"

FIFO_PATH="$FIFO_DIR/$$"
mkfifo "$FIFO_PATH"

cleanup() {
    rm -f "$FIFO_PATH"
    if [ -f "$LOCK_FILE" ] && [ "$(cat "$LOCK_FILE")" = "$$" ]; then
        rm -f "$LOCK_FILE"
        echo "Процесс $$ освободил ресурс при завершении"
        next_fifo=$(ls "$FIFO_DIR" | sort -n | head -n 1)
        if [ -n "$next_fifo" ]; then
            echo "1" > "$FIFO_DIR/$next_fifo"
        fi
    fi
    exit
}

trap cleanup EXIT SIGINT SIGTERM

echo "Процесс $$ ожидает освобождения ресурса..."
```

Рис. 4.4: new semaphor.sh

```

if [ ! -f "$LOCK_FILE" ]; then
    echo "$$" > "$LOCK_FILE"
    echo "Процесс $$ получил доступ к ресурсу (ресурс был свободен)"
else
    echo "Процесс $$ встал в очередь на ресурс"
    while read -r; do
        if [ ! -f "$LOCK_FILE" ]; then
            echo "$$" > "$LOCK_FILE"
            echo "Процесс $$ получил доступ к ресурсу (получил уведомление)"
            break
        fi
    done < "$FIFO_PATH"
fi

echo "Процесс $$ использует ресурс..."
sleep $T2

rm -f "$LOCK_FILE"
echo "Процесс $$ освободил ресурс"

next_fifo=$(ls "$FIFO_DIR" | sort -n | grep -v "^$$" | head -n 1)
if [ -n "$next_fifo" ]; then
    echo "1" > "$FIFO_DIR/$next_fifo"
fi

rm -f "$FIFO_PATH"

```

Рис. 4.5: new semaphor.sh

Дорабатываем программу так, чтобы имелась возможность взаимодействия трёх и более процессов.

```

[adpetlin@adpetlin ~]$ ./semaphor.sh > /dev/tty2
[adpetlin@adpetlin ~]$ |

```

Рис. 4.6: работа скрипта

Перенаправляем вывод из первого терминала во второй.

```

[adpetlin@adpetlin ~]$ ./semaphor.sh > /dev/tty3
[adpetlin@adpetlin ~]$ |

```

Рис. 4.7: работа скрипта

Перенаправляем вывод из второго терминала в третий.

```
[adpetlin@adpetlin ~]$ ./semaphor.sh
Процесс 17019 ожидает освобождения ресурса...
Процесс 17019 получил доступ к ресурсу (ресурс был свободен)
Процесс 17019 использует ресурс...
Процесс 17019 освободил ресурс
[adpetlin@adpetlin ~]$ |
```

Рис. 4.8: работа скрипта

Вывод в третьем терминале

```
#!/bin/bash

MAN_DIR="/usr/share/man/man1"
COMMAND=$1
MAN_FILE="$MAN_DIR/${COMMAND}.1.gz"

if [ -f "$MAN_FILE" ]; then
    gunzip -c $MAN_FILE | less
else
    echo "Справка по команде '$COMMAND' не найдена."
fi
```

Рис. 4.9: reman.sh

Реализовываем команду man с помощью командного файла.

```

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH PWD "1" "November 2024" "GNU coreutils 9.5" "User Commands"
.SH NAME
pwd \- print name of current/working directory
.SH SYNOPSIS
.B pwd
[\fI\,OPTION\ \fR]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Print the full filename of the current working directory.
.TP
\fB\-\fR, \fB\-\-logical\fR
use PWD from environment, even if it contains symlinks
.TP
\fB\-\fR, \fB\-\-physical\fR
avoid all symlinks

```

Рис. 4.10: работа скрипта

Вывод для указанного аргумента “pwd”.

```

#!/bin/bash

LENGTH=${1:-10} # Длина последовательности (по умолчанию 10)

for ((i=0; i<LENGTH; i++)); do
    RAND_NUM=$((RANDOM % 26))
    LETTER=$(printf \"\$(printf '%03o' $((97 + RAND_NUM)))\"
    echo -n "$LETTER"
done
echo

```

Рис. 4.11: randoml.sh

Используя встроенную переменную \$RANDOM, пишем командный файл, генерирующий случайную последовательность букв латинского алфавита.

```
[adpetlin@adpetlin ~]$ chmod +x random1.sh
[adpetlin@adpetlin ~]$ ./random1.sh
tyrnowzsnv
[adpetlin@adpetlin ~]$ ./random1.sh 5
nuvsj
[adpetlin@adpetlin ~]$ ./random1.sh 20
grbwforiotkfirwfmqct
[adpetlin@adpetlin ~]$ ./random1.sh 40
cpdnixnyfybnsnmejsmjrmgrwjbddjffjmdspalv
[adpetlin@adpetlin ~]$ |
```

Рис. 4.12: работа скрипта

Вывод при разных аргументах или их отсутствии.

5 Выводы

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

Список литературы

1. Dash, P. Getting Started with Oracle VM VirtualBox / P. Dash. – Packt Publishing Ltd, 2013. – 86 сс.
2. Colvin, H. VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox. VirtualBox / H. Colvin. – CreateSpace Independent Publishing Platform, 2015. – 70 сс.
3. Vugt, S. van. Red Hat RHCSA/RHCE 7 cert guide : Red Hat Enterprise Linux 7 (EX200 and EX300) : Certification Guide. Red Hat RHCSA/RHCE 7 cert guide / S. van Vugt. – Pearson IT Certification, 2016. – 1008 сс.
4. Робачевский, А. Операционная система UNIX / А. Робачевский, С. Немнюгин, О. Стесик. – 2-е изд. – Санкт-Петербург : БХВ-Петербург, 2010. – 656 сс.
5. Немет, Э. Unix и Linux: руководство системного администратора. Unix и Linux / Э. Немет, Г. Снайдер, Т.Р. Хейн, Б. Уэйли. – 4-е изд. – Вильямс, 2014. – 1312 сс.
6. Колисниченко, Д.Н. Самоучитель системного администратора Linux : Системный администратор / Д.Н. Колисниченко. – Санкт-Петербург : БХВ-Петербург, 2011. – 544 сс.
7. Robbins, A. Bash Pocket Reference / A. Robbins. – O'Reilly Media, 2016. – 156 сс.