ЛАБОРАТОРНАЯ РАБОТА №2

МУЛЬТИЗАДАЧНЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Программирование на языке командного интерпретатора SHELL.

Цель лабораторной работы.

Целью данной лабораторной работы является ознакомление с основными возможностями, предоставляемыми языком программирования командных последовательностей shell. Основная практическая цель – выполнение индивидуальных заданий в рамках сеанса работы с ОС UNIX, смысл которых состоит в выполнении процессов пользователя в фоновом режиме, в перенастроенной среде, в выполнении принудительного завершения процессов пользователя, а также в создании `.profile`-файла и файлов командных последовательностей.

КОМАНДНЫЙ ИНТЕРПРЕТАТОР SHELL.

Командный интерпретатор shell не входит в ядро операционной системы UNIX, что позволяет пользователю во-первых, выбирать такой командный интерпретатор, который в наибольшей степени удовлетворял бы запросам пользователя, а во-вторых, создавать свои собственные программы на языке shell. С точки зрения операционной системы командный интерпретатор shell обеспечивает удобный интерфейс между пользователем и ядром с помощью выполнения вводимых с терминала команд. Кроме того, командный интерпретатор shell (чаще всего используется bourne shell) – предоставляет пользователю полноценный язык программирования с возможностью введения переменных, алгоритмических структур и подпрограмм (содержащих командные файлы в том числе), обеспечением возможности передачи параметров и обработки прерываний.

Программы (или скрипты), написанные на языке shell, представляют из себя последовательность командных строк, предназначенных для обработки интерпретатором языка shell. Сам интерпретатор обычно расположен в директории `/bin` под именем `sh`.

Командная строка в языке shell – это последовательность слов, разделенная пробелами, причем всегда первое слово определяет имя команды, утилиты или shell-программы, которые будут выполнены, а оставшиеся слова, как правило, передаются команде в качестве аргументов. Файл, содержащий программу, определяемую первым словом строки, может храниться как в системном каталоге, так и в личных каталогах и быть продуктом деятельности пользователя. Таким образом, набор команд языка shell является расширяемым – его пополняет каждая программа, написанная пользователем, и даже ссылка на уже существующий файл. Для процедур shell остаются таким образом в силе понятия «стандартный ввод», «стандартный вывод», «стандартный протокол» и «конвейер». Кроме того, мощь языку shell придаёт возможность использования переменных и аргументов. Переменные в языке shell обозначаются идентификаторами, а определяются строками типа:

```
var=value;
export var;
```

где var – соответствующий текстовому значению value идентификатор. Доступ к переменной (к её значению) осуществляется по выполнению по отношению к идентификатору префиксной операции $. Строка, являющаяся значением (value) некоторой переменной языка shell, не должна содержать символы переопределения ввода/вывода, фонового процесса и канала. Предопределенные (служебные) переменные shell состоят из двух подмножеств:

аргументов процедур и внутренних переменных идентификатора, некоторые из которых можно только читать, а другие – ещё и модифицировать.

К внутренним shell-переменным относятся такие, например, как:

HOME - содержит имя "домашнего" каталога пользователя.

PATH - множество каталогов, в которых будет проходить поиск команды. Разделитель – «:».

PS1 и / или **PS2** - первичная и вторичная подсказки системы ("промптеры").

TERM - содержит мнемоническое имя типа терминала пользователя.

MAIL - содержит полное имя файла почтового ящика
пользователя.

Аргументы – это особый тип переменных, именами которых служат цифры. Аргументы – это по сути своей позиционные параметры командной строки, причем имя команды – это нулевой аргумент. Значения аргументов обозначают как **$d**,

где **d** - любая десятичная цифра. Поскольку в некоторых случаях десяти переменных может оказаться недостаточно, то в языке shell введена команда shift, сдвигающая аргументы на одну позицию влево за исключением **$0**, который остается без изменений.

Значения некоторых внутренних переменных (так называемых "зарезервированных") устанавливаются самим shell'ом, а не пользователем:

- количество фактических аргументов – позиционных параметров, кроме **$0**,

? - код возврата,

$ - значение PID (см. лабораторную работу 1) текущего процесса,

! - значение PID фонового процесса, вызванного последним когда-либо,

- - состояние флагов интерпретатора shell,

***** - совокупность всех позиционных параметров, начиная с **$1**.

Для выполнения арифметических действий используется команда expr, помещающая результат в стандартный вывод.

Знаки операций команды expr:

+ - для сложения,

- - для вычитания,

'*' или **"*"** - для умножения,

/ - для деления,

% - для взятия остатка от деления.

Язык shell кроме того, как всякий универсальный язык программирования предоставляет ряд конструкций для управления последовательностью действий:

- оператор цикла for;
- условный оператор if;
- команда test для вычисления условных выражений;
- оператор цикла while;
- оператор цикла until;
- оператор выбора case;
- операторы выдачи сообщений echo и ewal.

Наиболее часто используются операторы цикла и условный оператор, при этом формат оператора цикла for можно представить в виде:

for <имя> [**in** <слово>...]
do
<командная последовательность>
done

где переменная <имя> последовательно принимает значения из набора <слово>, или, в случае отсутствия конструкции in <слово>, соответствует значению формального аргумента;

do... done - тело цикла; <командная последовательность> - любая последовательность команд shell'a.

Построение циклов возможно и с помощью следующих форматов:

while <условие>
do
<командная последовательность>
done

где <условие> - некоторая переменная, константа или выражение, вырабатывающее код возврата. При значении кода возврата <условие> равного нулю будет выполняться тело цикла.

until <условие>
do
<командная последовательность>
done

В этом формате тело цикла исполняется при ненулевых значениях кода возврата <условие>.

Условный оператор имеет несколько форматов, при этом одним из наиболее употребительных является следующий:

if <условие>
then
<командная последовательность 1>
[**else**
<командная последовательность 2>]
fi

где <командная последовательность 1> выполняется в случае выработки в <условие> кода возврата, равного нулю, после чего выполняется, если она есть, команда, непосредственно следующая за ограничителем fi; <командная последовательность 2> выполняется сразу после выработки в <условие> ненулевого кода возврата.

Язык командного интерпретатора shell дает большие возможности пользователю по созданию гибкого программного окружения. Другими распространенными версиями shell являются т.н. C-shell (автор - У.Джой) и Korn-shell (автор - Д. Корн), каждый из которых при идейной близости к bourne-shell обладает отличительными особенностями только в пла не улучшения сервисных служб (введение буфера диалогового режима, массива переменных и т.д.), в связи с чем изучение языков командного интерпретатора в ОС UNIX происходит, как правило, на примере Bourne-shell'a.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Домашняя подготовка

1. Выбрать для изучения вариант языка командного интерпретатора Bourne-shell.
2. Изучить по прилагаемому списку литературы следующие вопросы и темы: файлы инициализации; переменные и аргументы; операторы и процедуры; построение условных операторов, циклов; выполнение проверок, сдвигов, ветвления, вычислений; замены, подготовки и переназначения.

3. Для изучивших Bourne-shell ответить на следующие вопросы коллоквиума (изучающие C-shell отвечают на те же вопросы, но заменяют в тексте вопроса "Bourne-shell" на "C-shell"):

1) Как определен в Bourne-shell файл начала сеанса?

2) Каким образом строится процедура в Bourne-shell?

3) Каков порядок поиска по каталогам ОС типа UNIX при использовании bourne-shell?

4) Как называются командные файлы в bourne-shell?

5) Каким образом устанавливается в bourne-shell первичная подсказка-запрос системы?

6) Каким образом устанавливаются каталоги обязательного поиска команд в bourne-shell?

7) Каким образом устанавливается средствами bourne-shell тип и характеристики терминала пользователя?

8) Каким образом устанавливается в bourne-shell вторичная (собственно bourne-shell'овская) подсказка-промптер?

9) Какие типы переменных и аргументов доступны в bourne-shell?

10) Какие возможности группирования команд Вам известны в Bourne-shell, каким образом они определены для интерпретатора?

11) Что такое shell-файл в Bourne-shell?

12) Каким образом выполнить Shell-файл в Bourne-shell?

13) Каким образом осуществляется вывод любого готового текста на экран пользователя Bourne-shell'a?

14) Каким образом может быть осуществлена проверка файлов в Bourne-shell?

15) Каким образом может быть произведено построение условных операторов в Bourne-shell?

16) Каким образом получить в Bourne-shell аргумент с номером i?

17) Каким образом получить в Bourne-shell идентификационные номера процессов и каких процессов?

18) Каким образом присваивается переменная в Bourne-shell?

19) Каким образом отображается (возвращается) в Bourne-shell значение переменной?

20) Можно ли присваивать переменным Bourne-shell'a текстовые значения, содержащие разделители (например, пробелы)?

21) Каким образом происходит вычисление арифметических выражений в Bourne-shell?

22) Каким образом Bourne-shell'у доступно число аргументов команды?

23) Каким образом может быть произведено построение операторов цикла в Bourne-shell?

24) Возможно ли и каким образом построение вложенных циклов в Bourne-shell?

25) Возможно ли принудительное завершение цикла в Bourne-shell?

26) Возможно ли возобновление выполнения цикла в Bourne-shell?

27) Возможно ли принудительное завершение процедуры Bourne-shell'a?

28) Каким образом можно построить многоальтернативный выбор в Bourne-shell?

29) Каким образом получить в Bourne-shell доступ к аргументам некоторой команды, если число таких аргументов более 10?

30) Каким образом в Bourne-shell производится обработка ситуаций, вызывающих выработку сигналов прерываний?

4. Просмотрите лабораторное задание, соответствующее вашему варианту, продумайте его выполнение.

Лабораторное задание

1) Перед выполнением лабораторного задания :

- ответьте на вопросы коллоквиума, указанные преподавателем;

- получите у администратора системы или преподавателя право на использование терминала;
- зарегистрируйтесь в системе.

2) Создайте в вашем домашнем каталоге подкаталог lab4, перейдите в него.
3) Сохранить в виде ascii- файла F1.var значение внутренних переменных shell и номера процессов в системе.
4) Переопределить значения переменных PS1, PS2, сохранить в ascii- файле F2.var значения внутренних переменных.
5) Вернуться к исходному значению внутренних переменныхshell.
6) В каталоге lab4 получить копию файла

/ete/skel/local.profile

с именем .profile, определить в нем значения переменных DIGIT_FOR_LAB, MYOWNNAME, TIMEZONE, MYGROUP, DAY_OF_WEEK_TODAY, а также измените на действующие значения внутренних переменных интерпретатора PATH и HOME.

Стартуйте модифицированную таким образом командную оболочку. Убедитесь в результате.

7) Напишите программу на языке shell - "скрипт", которая по вводимой вами произвольной последовательности десятичных чисел осуществляла бы вычисление промежуточных сумм и выводила конечный результат на экран и в файл sh.result. Используйте в программе значение переменной интерпретатора S#. Если номер Вашей бригады четный - постарайтесь обойтись оператором for, если нечетный - конструкцией if.
8) Напишите скрипт, который выводил бы через каждые десять секунд на экран некоторое сообщение только в том случае, если наступит определенное время в определенный день. Режим запуска - фоновый. Уточнение задания - см. табл.4.1
9) Напишите скрипт, который проверял бы наличие процесса, связанного со скриптом по пункту номер 8) и останавливал бы его, если количество сообщений, выведенных на экран скриптом по пункту номер 8) превысит некоторое число. Уточнение задания - см. табл.4.2.
10) Напишите скрипт без использования диалога с оператором на языке shell, который через несколько минут (5-10) закроет все процессы, связанные с пунктами 7 - 9. Если номер Вашей бригады четный - постарайтесь обойтись конструкцией if, иначе - конструкцией for. Примечание: перед запуском скрипта покажите его текст преподавателю и только после получения разрешения - запустите на исполнение. Подсказка к выполнению: кое-о-чем Вы позаботились в начале лабораторного задания; это "кое-что" позволит Вам выполнить задание.
11) После запуска скрипта п.10 вернитесь в оболочку shell которая была в Вашем распоряжении до выполнения п.6).
12) Завершите сеанс работы с ОС через минуту запуска некоторого скрипта.

ТАБЛИЦА 4.1

№ бригады	Уточнение задания к п.8)
	Время срабатывания скрипта "запомнить" в переменных DAY_OF_WEEK_TODAY, TIMEZONE
	Аргумент при запуске скрипта – значения для переустановки PS1
	После каждого вывода на экран сообщения значение переменной MYOWNNAME «удлинялось» бы на один символ из последовательности: A,B,C,D,E,F,....
	При выводе сообщения скрипт создает новую переменную MESSAGEn (n=1,...,N), значение которой состоит из номера сообщения и его текста
	Сообщение представляет собой следующий текст: - строка заголовка,

	- порядковый номер текущего сообщения, - номер PID скрипта и его PPID, - время, - имя процесса из активных в этот момент с его PID.
	Сообщение представляет собой следующий текст: - строка заголовка - **** - " "x" , где x - значение MYOWNNAME - ' 'y' , где y - значение DAY_OF_WEEK_TODAY - $$$z$# , где z - значение MYGROUP
	Сообщение представляет собой следующий текст: - строка заголовка - текущее системное время - "звонок" - список всех пользователей, у которых активны процессы с PID>=PID скрипта по п.8)
	Запуск скрипта выполните в последовательности с командой date, строка сообщения должна включать в себя количество секунд, прошедших с начала запуска скрипта.

ТАБЛИЦА 4.2

№	Уточнение задания к п.9)
1 - 8	Необходимое число повторов - в переменной DIGIT_FOR_LAB
1,3,5,7	Значение DIGIT_FOR_LAB установить в диалоговом режиме после старта скрипта п.9).
2,4,6,8	Значение DIGIT_FOR_LAB установить на N больше, чем количество повторов по скрипту п.8), установка N - в диалоговом режиме.
1,2,3,4	Значение DIGIT_FOR_LAB не может быть отрицательным, нулевым и больше 5.
5,6,7,8	Значение DIGIT_FOR_LAB может быть только одним из следующих: 2,3,5,7,10,100,120.
1 - 8	Первое сообщение после старта скрипта по п.9) - вывод значения текущего числа повторов сообщений, выведенных скриптом по п.8).