Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА
(МГТУ им. Н.Э. Баумана)
Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Галямова Е. В., Абанин И. С.

Лабораторная работа № 6

Системные сервисы z/OS UNIX

Методические рекомендации по курсу
«ОПЕРАЦИОННАЯ СИСТЕМА Z/OS и ВИРТУАЛЬНАЯ СРЕДА MAINFRAME»

2007 год,
Москва

Лабораторная работа № 6

Работа с системными сервисами UNIX операционной системы z/OS

Цель лабораторной работы:

- знакомство с основами использования системных сервисов операционной системы большой вычислительной машины z/OS UNIX;
- освоение структуры иерархической файловой системы HFS (Hierarchic File System) операционной системы z/OS и основных команд для работы с ней;
- изучение командной оболочки shell;
- изучение скриптового языка awk;
- создание простой С программы;

1. *Теоретическая часть*

1.1 .Основы z/OS UNIX.

В z/OS Unix реализовано два открытых системных интерфейса:

- интерфейс системных вызовов API
- интерактивный интерфейс пользователя.

Интерфейс системных вызовов (API) позволяет запускать стандартные Unix приложения, написанные на C, в z/OS. Интерактивный интерфейс пользователя позволяет (shell) позволяет выполнять Unix команды, утилиты и скрипты в z/OS.

Ядро z/OS UNIX интегрировано в базовую управляющую программу z/OS и служит для реализации функций интерфейса системных вызовов (API UNIX), связанных с управлением процессами, файловой системой HFS и коммуникациями. Активизируется при загрузке z/OS и работает в собственном адресном пространстве MVS.

Основная единица работы в операционной системе UNIX – процесс, он соответствует находящейся в стадии выполнения программе со всеми выделенными ей ресурсами. Процессы выполняются исключительно в адресных пространствах MVS. При использовании системного вызова fork() всегда создается новое адресное пространство, являющееся копией родительского. При использовании системного вызова spawn() может быть как создано новое адресное пространство, так и запущена новая задача внутри родительского AS.

Процессы: бывают системные (например, демоны - работающие в фоновом режиме и предназначенные для поддержки вспомогательных системных сервисов - вывод на печать, электронная почта, запуск программ по расписанию и т. д.) и пользовательские.

Каждый процесс имеет уникальный идентификатор PID и может по своей инициативе порождать новые (дочерние) процессы с помощью системных вызовов fork() и spawn(). Также каждый процесс имеет родительский процесс, который определяется идентификатором родительского процесса PPID (см. рисунок 1).

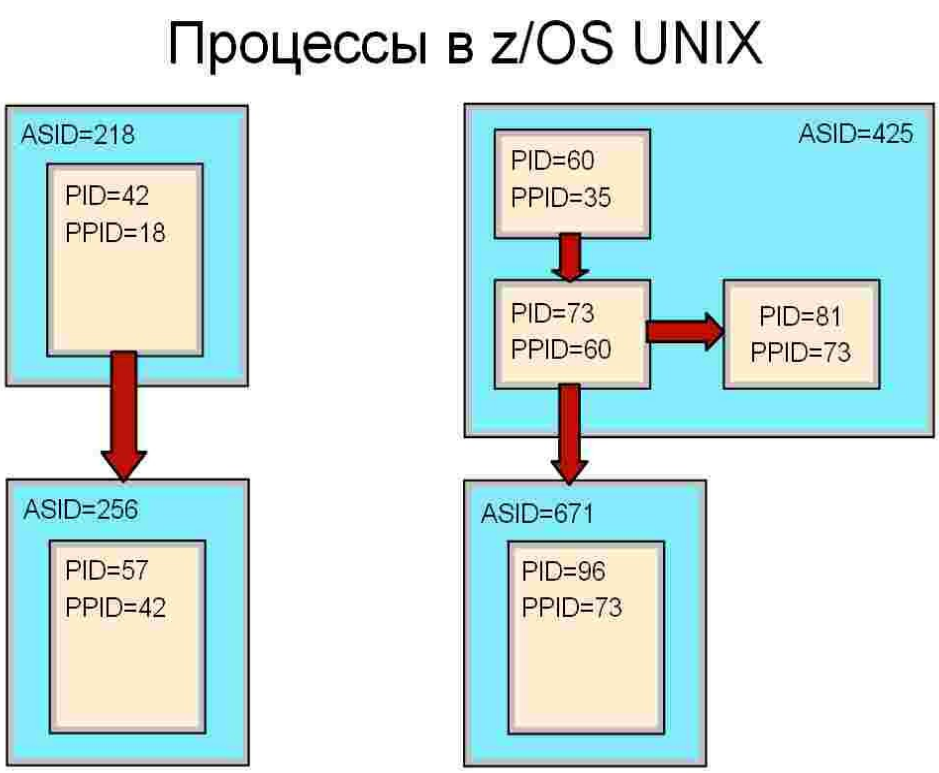

Рисунок 1 – Процессы в z/OS UNIX.

Механизм выполнения приложения UNIX в z/OS показан на рисунке 2. Для ядра z/OS UNIX во время инициализации системы создается отдельное адресное пространство (OMVS), функционирование которого зависит от настроек в разделе BPXPRMxx системного реестра SYS1.PARMLIB. Одновременно с этим создается файловая система HFS (подробнее о файловой системе см. в п.2) и создается адресное пространство BPXOINIT, выполняющее процесс прародитель (PID = 1) для всех процессов. В первую очередь BPXOINIT порождает необходимые системные процессы UNIX.

Рисунок 2 – Механизм выполнения приложения UNIX в z/OS.

Новые адресные пространства для приложений UNIX создаются по запросу ядра к менеджеру управления рабочей нагрузкой WLM. WLM использует для создания нового адресного пространства специальную STC-процедуру BPXAS.

Как правило, для каждого интерактивного пользовательского сеанса командные интерпретаторы shell запускаются в отдельных адресных пространствах. Команды и утилиты пользователя могут запускаться как внутри адресного пространства shell, так и в новых адресных пространствах.

Процесс-демон INETD обеспечивае доступ к shell для удаленных пользователей в TCP/IP-сети с использованием протоколов telnet и rlogin.

1.2 .Иерархическая файловая система HFS.

Файлы UNIX обрабатываются системой как простая совокупность байтов, без деления на логические записи. Имена файлов могут содержать до 255 алфавитно-цифровых символов, при этом различают прописные и строчные буквы.

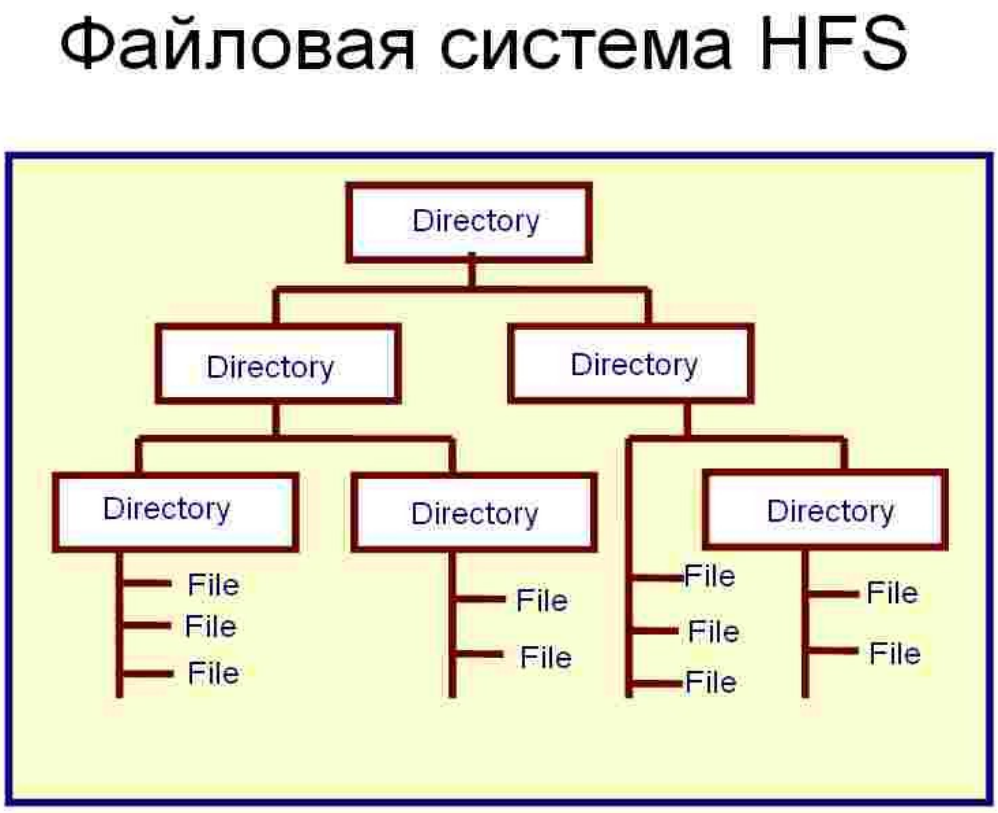

Рисунок 3 – Иерархическая файловая система.

В UNIX выделяют следующие типы файлов:

- обычные - файлы общего назначения, используемые для хранения программ и данных любого типа;
- каталоги - служат для размещения справочной информации о размещении файлов, принадлежащих данному каталогу;
- устройства - ассоциируются с устройствами ввода-вывода;
- символические ссылки - содержат ссылки на другие файлы;
- именованные каналы - служат для обмена данными между процессами;
- сокеты - служат для реализации сетевого взаимодействия.

Для работы с файлами и каталогами в z/OS UNIX используется командная оболочка shell. С помощью специальных команд вы можете создавать и удалять каталоги и файлы, изменять права доступа, работать с жесткими и символическими ссылками.

Файлы группируются по директориям, образуя иерархическую древовидную структуру

(см. рисунок 3). Вершиной дерева и единой точкой входа в файловую систему является корневой каталог (/). Таким образом, у каждого файла существует полное или абсолютное имя, однозначно определяющее его местоположение в файловой системе: /u/user1/docs/abc, /u/user2/prg и т.п. Наиболее важные системные программы, данные и конфигурационные файлы UNIX размещаются в специальных каталогах: /bin - команды и утилиты; /usr - файлы для поддержки решения пользовательских задач; /dev - специальные файлы устройств ввода-вывода; /etc - утилиты администрирования и конфигурационные файлы; /lib - включаемые библиотеки C/C++; /tmp - временные файлы; /var - сообщения и системные журналы; /samples - примеры программ и настроечных файлов.

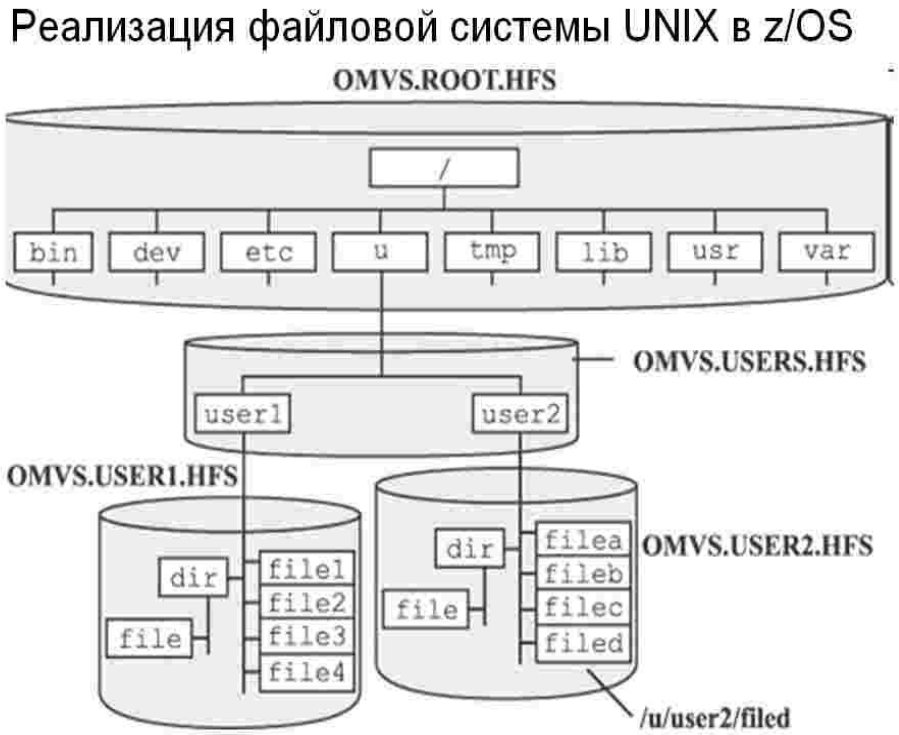

Рисунок 4 – Реализация иерархической файловой системы.

Вершиной дерева и единой точкой входа в файловую систему является корневой каталог (/). Таким образом, у каждого файла существует полное или абсолютное имя, однозначно определяющее его местоположение в файловой системе: /u/user1/docs/abc, /u/user2/prg и т.п. Наиболее важные системные программы, данные и конфигурационные файлы UNIX размещаются в специальных каталогах: /bin - команды и утилиты; /usr - файлы для поддержки решения пользовательских задач; /dev - специальные файлы устройств ввода-вывода; /etc - утилиты администрирования и конфигурационные файлы; /lib - включаемые библиотеки C/C++; /tmp - временные файлы; /var - сообщения и системные журналы; /samples - примеры программ и настроечных файлов.

Все особенности иерархической файловой системы HFS поддерживаются системными сервисами UNIX в z/OS.

Для размещения файлов UNIX и реализации иерархической структуры доступа создаются специальные однотомные SMS-управляемые наборы данных, получившие название наборов данных HFS (см. слайд). Каждый набор данных HFS содержит определенный сегмент файловой системы, точкой входа в который является один из каталогов.

Объединение сегментов HFS производится с помощью специальной операции "монтирования", выполняемой на этапе инициализации системы или динамически. Первым всегда монтируется сегмент, содержащий корневой каталог файловой системы (/), к которому затем могут добавляться другие сегменты. Создание и управление наборами данных HFS осуществляется стандартным компонентом z/OS DFSMS.

1.3 .Скриптовый язык awk

Утилита AWK предназначена для обработки данных, содержащихся в файлах. Она включает в себя простой скриптовый язык, позволяющий создавать простые программы для обработки текстовых данных. AWK сканирует входные текстовые файлы и обрабатывает строки, удовлетворяющие определенным в программе условиям. Максимальный размер строки 256 символов.

AWK программа состоит из следующих компонентов: полей данных, стандартных переменных, массивов, паттернов и действий. Паттерн состоит из регулярных выражений, выражений отношений и комбинаций паттернов. А также ключевых слов BEGIN и END. Внутри BEGIN { } определяется набор действий, которые необходимо выполнить до начала обработки файлов. Внутри END { } указывается набор действий, выполняемый после обработки входных данных. Действие – последовательность выражений, разделенных ; .AWK поддерживает стандартный набор арифметических операций. Каждая строка входного файла обрабатывается как совокупность полей и разделительных символов (по умолчанию пробелов). Для обращения к полю с номером n достаточно написать $n. Переменные и массивы принимают значения в зависимости от контекста.

Для запуска AWK программы необходимо ввести следующую команду:

```
Awk –f <имя файла с AWK программой > <имя обрабатываемого файла>
```

1.4 .Режимы доступа пользователей к z/OS UNIX

Для пользователей z/OS UNIX поддерживается несколько различных режимов интерактивного доступа к системным сервисам UNIX, как с помощью shell, так и некоторыми другими способами.

Первый режим подключения (терминал пользователя через TCP/IP соединение) является традиционным для пользователей z/OS и основан на использовании компонентов TSO/E и ISPF. В TCP/IP-сети терминалы TSO поддерживаются на основе специального протокола TN3270, представляющего собой адаптированный вариант стандартного протокола telnet.

Второй режим доступа к сервисам UNIX основан на использовании стандартных прикладных протоколов TCP/IP rlogin и telnet. Для работы используется асинхронный ввод, возможно использование текстового редактора vi, но ограничен доступ к командам TSO. Также необходима предварительная настройка серверных компонент telnet и rlogin, а также разрешений на доступ в профиле RACF пользователя.

Третий режим доступа к сервисам UNIX основан на использовании ftp-протокола, также являющегося стандартным прикладным протоколом TCP/IP. В данном режиме можно получать доступ к данным MVS и UNIX.

2 .Практическая часть

2.1 .Запуск оболочки shell системных сервисов UNIX

Запустите эмулятор терминала 3270 и подключитесь к TSO, используя логин TSO ID, выданный Вам преподавателем. Введите пароль и запустите интерфейс системных сервисов OS/390 UNIX shell. Если процедура подключения не вызвала интерфейс ISPF, вы должны увидеть командную строку “TSO READY”. Если же процедура подключения вызвала ISPF, то вы увидите экран, показанный на рисунке 6.

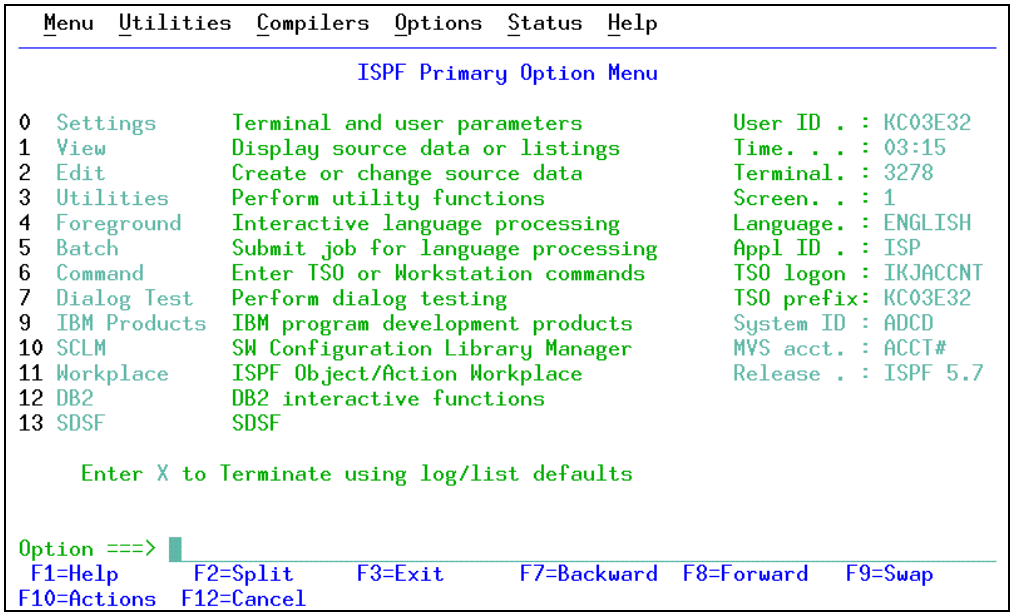

Рисунок 6 - Главное меню ISPF.

Если вы видите строку TSO READY, введите команду omvs, чтобы вызвать оболочку системных сервисов UNIX - USS shell. Если же вы находитесь в главном меню ISPF, то вы можете либо выйти из ISPF в TSO и затем ввести команду omvs, или же выбрать раздел 6 меню ISPF и в нем ввести команду omvs. Содержимое экрана должно быть таким, как показано на рисунке 7.

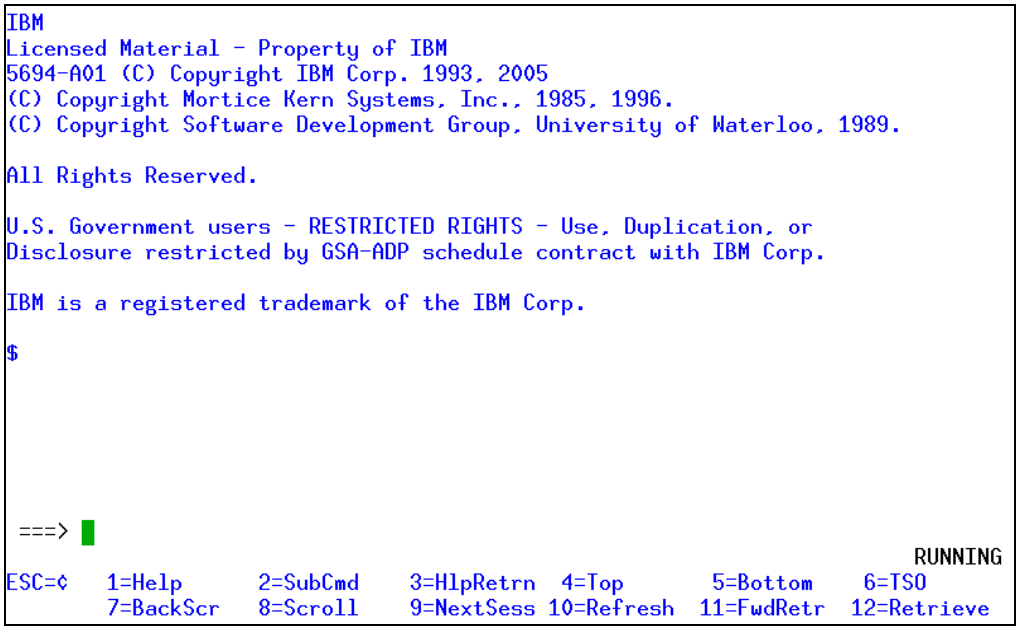

Рисунок 7 – Окно приветствия оболочки shell

Введите команду для просмотра вашего текущего рабочего каталога pwd. Посмотрите, как называется ваш рабочий каталог и зафиксируйте это. Найдите корневой каталог.

Введите команду для вывода всех файлов вашего текущего рабочего каталога

```
ls -a
```

Сколько элементов было выведено в результате предыдущей команды?

***Замечание*:** Если не был выведен ни один элемент, то это потому, что вы не указали опцию -a, в качестве параметра команды ls.

Введите команду для перехода в родительский каталог вашего домашнего

каталога -

```
cd /u
```

Посмотрите, является ли он вашим домашним каталогом, если нет, то как называется родительский каталог вашего домашнего каталога.

Еще раз введите команду для вывода имени текущего рабочего каталога

```
pwd
```

Как теперь называется ваш текущий рабочий каталог? Является ли этот каталог вашим домашним каталогом? Изменился ли ваш домашний каталог?

Выведите все файлы данного каталога вместе с их атрибутами

```
ls -al
```

Посмотрите, к какому типу относится большинство файлов данного каталога, как выглядит строка разрешений, присвоенная вашему домашнему каталогу, как эти разрешения будут выглядеть в восьмеричном представлении и показан ли на экране номер inode для этих файлов.

Замечание: Если результаты команды ls –al не помещаются на экран, и экран автоматически прокручивается, используйте PF клавиши F7 и F8 для прокрутки экрана вручнуют.

Введите команду для вывода номера inode вместе с другой информацией из предыдущего вопроса. Команда: ls –ali. Посмотрите, какой inode номер у вашего домашнего каталога.

Существует множество способов для перехода в ваш домашний каталог, это команды:

```
cd /u/xxxxxx
cd xxxxxx
cd $home
cd .
cd
```

Замечание: Где xxxxxx ваш TSO user id в нижнем регистре.

Вернитесь в ваш домашний каталог, используя одну из перечисленных выше команд.

2.2 .Работа с файлами и каталогами

2.2.1 .Создание и копирование файлов и каталогов

Создайте новый каталог с именем subdir1 из вашего домашнего каталога. Для этого используйте команду

```
mkdir subdir1
```

Из вашего домашнего каталога создайте еще один каталог под названием subdir2.

Перейдите в каталог subdir1. Создайте в нем файл hobbies и запишите в него список файлов из вашей домашней директории и вернитесь в домашний каталог.

Команды:

```
cd subdir1
touch hobbies
ls –l > hobbies
```

Просмотрите содержимое файла hobbies. Для этого используйте команду

```
cat hobbies
```

Скопируйте файл hobbies из каталога /u/xxxxxx/subdir1/ в ваш домашний каталог. Не изменяйте ваш текущий рабочий каталог.

```
cp /u/xxxxxx/subdir1/hobbies /u/xxxxxx/
```

Перечислите атрибуты вашего нового файла вместе с номером inode. Для этого используйте команду

```
ls -li hobbies
```

Посмотрите, какие разрешения установлены для созданного вами нового файла hobbies. Они должны быть следующими: -rw-rw-rw-.

Выведите содержимое файла hobbies

```
cat hobbies
```

Создайте файл comics.lst в каталоге subdir2.

Команды:

```
cd subdir2
touch comics.lst
ls –li > comics.lst
cd $home
```

Скопируйте файл comics.lst из каталога /u/xxxxxx/subdir2 в ваш каталог subdir1.

```
cp /u/xxxxxx/subdir2/comics.lst /u/xxxxxx/subdir1
```

Переименуйте ваш подкаталог subdir2 в links.

```
mv subdir2 links
```

Выведите содержимое файла comics.lst.

```
cd subdir1
cat comics.lst
```

Чтобы при выводе содержимого файла его содержимое последовательно выводилось на экран, имеется команда:

```
more comics.lst
```

Создайте файл под названием mtfile в вашем домашнем каталоге, используя команду touch, затем выведите атрибуты этого файла. Обратите внимание, что размер этого файла zero (0) bytes.

Введите команду touch для файла hobbies в вашем домашнем каталоге. Команда touch обновляет последнюю измененную дату и timestamp существующего файла или она создает новый пустой файл.

Подсказка: Посмотрите на время последнего обновления файла.

Введите команду, которая выведет все файлы из вашего домашнего каталога и перенаправить вывод в файл myfiles.

```
ls -l > myfiles
```

Добавьте текущие время и дату к содержимому файла myfiles.

```
date >> myfiles
```

Выведите содержимое файла myfiles.

```
cat myfiles
```

2.2.2 .Жесткие и символические ссылки

Создайте жесткую ссылку в каталоге links и назовите ее hlink. Эта ссылка должна указывать на файл comics.lst из каталога subdir1.

```
ln subdir1/comics.lst links/hlink
```

Выведите номера inode и остальные атрибуты файлов comics.lst и hlink.

```
ls -li subdir1/comics.lst links/hlink
```

Посмотрите, отличаются ли номера inode? Они должны быть одинаковыми, также как и остальные атрибуты (дата, размер, и т. д.). На эти файлы указывают две жесткие ссылки. Посмотрите, сколько сейчас существует файлов, содержащих данные из comics.lst .

Создайте символическую ссылку в каталоге links и назовите ее slink. Эта ссылка должна указывать на comics.lst из каталога subdir1.

```
ln -s /u/xxxxxx/subdirl/comics.lst /u/xxxxxx/links/slink
```

Выведите номера inode и другие атрибуты файлов comics.lst и slink.

```
ls -li subdir1/comics.lst links/slink
```

Как видите, не совпадают номера inode, также как и остальные атрибуты файлов (дата, размер и т.д.). На файл comics.lst указывают две жесткие ссылки и одна символическая. При этом, существует все еще только один файл с данными – comics.lst.

Примечание: Жесткая ссылка - это объект каталога, имеющий тот же номер inode, что и исходный файл, и являющийся по сути еще одним указателем на файл. Символическая ссылка – это файл, содержащий указатель на каталог исходного файла, а не на сам файл.

Создайте новый подкаталог в домашнем каталоге и назовите его newdir, переместите в него файл myfiles.

```
mkdir newdir
mv myfiles newdir
```

Выведите все файлы, подкаталоги и их содержимое, начинающиеся из вашего домашнего каталога.

```
ls –aR
```

Скопируйте файл comics.lst в каталог newdir.

```
cp subdir1/comics.lst newdir
```

2.2.3 .Удаление файлов, каталогов и ссылок

Догадайтесь, что произойдет при удалении файла comics.lst. Что произойдет с символической ссылкой? Ссылка останется, но будет указывать на несуществующий файл. Она будет отображаться при листинге, но открыть ее будет невозможно. А что произойдет с жесткой ссылкой? С жесткой ссылкой ничего не произойдет. Останется одна ссылка. Что при этом произойдет с данными на физическом уровне? С данными ничего не случится. Данные сохраняются до тех пор, пока существует хотя бы одна жесткая ссылка на них.

Удалите comics.lst и проверьте эти предположения.

```
rm subdir1/comics.lst
```

Что произойдет с символической ссылкой, если вы скопируете comics.lst обратно в каталог subdir1 из /u/xxxxxx/newdir? Символическая ссылка будет переназначена.

Что произойдет с жесткой ссылкой? С жесткой ссылкой ничего не произойдет. Новая

копия comics.lst будет иметь другой номер inode, то есть это будет другой файл.

Если вы скопируете comics.lst в ваш домашний каталог, то с символической ссылкой ничего не произойдет. Символическая ссылка будет указывать на каталог subdir1, а не на домашний каталог.

Скопируйте comics.lst обратно в каталог subdir1.

```
cp /u/xxxxxx/links/comics.lst subdir1
```

Попробуйте удалить каталог subdir2. Команда: rmdir newdir. Если каталог не пуст, то его нельзя удалить. Если бы этот каталог был не пуст, Вы могли бы удалить его, не удаляя все его файлы, просто воспользовавшись командой rm -r subdir2.

2.2.4 .Сортировка файлов

Отсортируйте содержимое файла hobbies из вашего каталога /subdir1, не используя опций команды.

```
sort /subdir1/hobbies
```

Записи в файле были отсортированы в алфавитном порядке, по первому символу каждой строки. При этом файл hobbies не изменяется. Отсортированные данные выводились только на экран.

Отсортируйте файл hobbies, используя второе поле каждой записи.

```
sort –k2 /subdir1/hobbies
```

Снова отсортируйте файл hobbies, используя второе поле каждой записи, но на этот раз укажите в параметрах команды игнорировать первый символ пробела в любом поле.

```
sort -b -k2 /subdir1/hobbies
или
sort -b +1 /subdir1/hobbies
```

Посмотрите, корректна ли сортировка на этот раз. Отсортируйте файл hobbies в соответствии с числами в последнем поле каждой записи.

```
sort -n -k4 /subdir1/hobbies
или
sort -n +3 /subdir1/hobbies
```

Отсортируйте файл comics.lst из вашего каталога subdir1. Отсортируйте второе поле файла, используя символ (:) в качестве разделителя полей.

```
sort -t: -k2 /subdir1/comics.lst
или
sort -t: +1 /subdir1/comics.lst
```

После выполнения этих команд файл был отсортирован по второму полю, но не в

числовом порядке, а в соответствии с кодом ASCII.

Отсортируйте этот файл снова, но в числовом порядке.

```
sort -t: -n -k2 /subdir1/comics.lst
```

2.3 .Изучение основных команд оболочки shell

Введите shell команду id. По этой команде система выводит номер uid, имя userid, gid номер и имя группы.

Введите команду date. Используйте TSO команду OBROWSE для просмотра файла hobbies или любого другого текстового файла из вашей домашней директории.

Введите команду tty. Данная команда выводит имя файла-терминала из каталога /dev, который представляет соединение терминала студента (с мейнфреймом).

Введите команду logname. Команда logname выводит logon имя пользователя.

Введите команду env и объясните значение выходных данных. Команда env выводит установленные переменные среды и их значения. Исходя из значений переменных среды, сколько строк текста будет выводиться на экран. Количество выводимых строк определяется переменной “lines”, обычно оно равно 20. Переменная для хранения домашнего каталога называется HOME.

Оболочка Shell расположена в /bin/sh. Если вы выведите информацию по команде date, используя команду type, будет выведено сообщение Date is cached /bin/date. Это означает, что команда date расположена в директории /bin, и что значение даты кэшируется в памяти для повышения производительности.

Просмотрите информацию о других известных вам командах.

Примечание: Для получения справки по синтаксису, параметрам и функциям shell команд используйте команду: man <имя команды>. Например, набрав “man cp” вы сможете увидеть информацию о команде cp. Для перемещения по страницам текста используйте клавишу Enter. Некоторые man описания достаточно длинные. Для выхода из справки man окна, нажмите q и вы вернетесь в режим командной строки.

Узнайте размер свободного пространства в смонтированной файловой системе, используя команду df. Посмотрите, чему равен размер свободного пространства в вашей файловой системе. Это значение меняется. Результат выводится в блоках (единицах).

Выведите содержимое каталога root с подробной информацией о его компонентах. Команда: ls -l / . Вы можете отличить каталог от файла, поскольку для каталога первый символ в выводимой строке “d”. Пятое выводимое поле означает размер компонента в байтах.

Выведите содержимое вашего домашнего каталога вместе с номерами inode.

```
ls –l /u/xxxxxx
```

Номер inode присваивается системой UNIX для управления файлами.

Выведите содержимое вашего домашнего каталога, перенаправив вывод в файл listout.

Команды:

```
cd $home
ls -li > listout
```

Воспользуйтесь командой cat для просмотра файла listout.

Создайте новый каталог в каталоге newdir и назовите его Dir1A.

```
mkdir newdir
mkdir newdir/Dir1A
```

Введите команду для просмотра битов разрешения вашего нового каталога.

```
ls -al newdir/DirlA
```

Биты разрешения для DirlA равны rwxr-xr-x или 755. Измените биты разрешений для данного каталога, так чтобы владелец каталога имел права чтения, записи и выполнения, члены группы и все остальные пользователи получили только право чтения.

```
chmod 744 newdir/DirlA
или
chmod go-x newdir/DirlA
```

Отсортируйте содержимое ранее созданного файла listout по значению поля “размер”. Перенаправьте вывод результатов в файл sorted. Поместите файл sorted в каталог Dir1A.

```
sort -k5 listout > newdir/DirlA/sorted
```

Объедините команду вывода содержимого каталога и команду сортировки и создайте конвейер команд, позволяющий просмотреть содержимое каталога в отсортированном по размеру порядке.

```
ls -l | sort -k5
```

Имейте в виду, что вы не сможете сделать это непосредственно в одной команде.

Используйте команду grep для вывода только тех записей файла hobbies, которые имеют отношение к “links” (поиск текста в файле).

```
grep links hobbies
```

Выведите содержимое каталога /etc. Обратите внимание на поле дата и убедитесь, что компоненты каталога отличаются по году или месяцу.

```
ls –l /etc
```

Объедините следующие задачи в одну командную строку, используя конвейер:

Выведите содержимое каталога /usr, выберите записи с определенным месяцем или днем месяца, отсортируйте выходные данные по размеру.

```
ls –l /usr | grep "Apr" | sort -k5
```

Выведите содержимое вашего домашнего каталога и перенаправьте вывод в файл outlist. Запустите эту команду в фоновом режиме.

```
ls -l > curlist &
```

В результате вы получите ID задачи (job id) и PID фонового процесса.

Повторите команду из пункта 22, запустив ее в фоновом режиме и перенаправив вывод в файл outlist.

```
ls -l /usr | grep "Apr" | sort -k5 > curlist &
```

Убедитесь, что была выведена одна ID задача (job ids) и три ID процесса. Имейте в виду, что id группы (GID) равен ID первого процесса.

Введите команду history. Эта команда выводит последние введенные 16 команд.

Выберите одну команду из истории команд и перезапустите ее, напечатав r перед номером, соответствующим выбранной команде.

```
r 173
```

Выведите содержимое файла sh_history.

```
cat .sh history
```

Файл .sh_history содержит список ранее введенных команд. Обратите внимание на разницу между списком команд, полученным при помощи команды history и содержимым файла .sh history. Команда history выводит последние 16 команд вместе с номерами, присвоенными этим командам. Файл .sh_history содержит гораздо больше команд, но не содержит их номера.

2.3.1 .Настройка оболочки shell

Введите команду echo для вывода значений переменных LOGNAME и PWD.

```
echo $LOGNAME $PWD
```

Переменная LOGNAME равна регистрационному имени пользователя. (logon name), а переменная PWD равна текущему рабочему каталогу.

Вы можете настроить оболочку shell таким образом, чтобы она выводила login и текущий рабочий каталог. Значение, присваиваемое переменной необходимо заключить в одинарные кавычки, тогда имя рабочего каталога будет корректно отображаться при вводе команды cd.

```
PS1 = '$LOGNAME:$PWD: >' `
```

Используя команду OEDIT, добавьте команду из предыдущего пункта к файлу .profile из вашего домашнего каталога. После завершения редактирования файла нажмите клавишу F3, чтобы его закрыть.

```
oedit .profile
```

2.3.2 .Написание shell скриптов

Создайте новый каталог scripts в вашем домашнем каталоге. Команда: mkdir scripts

Создайте новый файл myscript в каталоге scripts. Используя конвейер команд напишите команду для вывода содержимого текущей рабочей директории, отсортировав результаты по размеру, и сохраните данную команду в файле scripts.

Скрипт:

```
ls –l | sort –k5
touch  myscripts
ls –l | sort –k5 > myscripts
```

Запустите ваш новый скрипт. Команда: scripts/myscript. На экран будет выведено следующее сообщение: “Cannot execute, permission denied.” Если вы увидели это сообщение об ошибке, исправьте биты разрешения и перезапустите скрипт.

Команды:

```
chmod +x scripts/myscript
script/myscript
```

Напишите скрипт, спрашивающий у пользователя его имя и возраст и генерирующий предложение на основании полученных данных. Назовите его hello и запустите. Команда shell - read может считывать значение и присваивать его переменной. Команда echo может использоваться для вывода сообщения на экран.

Скрипт:

```
echo Please enter your name:
read name
```

```
echo Please enter your age:
read age
echo Hello, $name. You are $age years old.
```

Напишите скрипт, который делает следующее: запрашивает у пользователя его имя и год рождения; вычисляет возраст пользователя и выводит предложение содержащее этот возраст. Сохраните файл под именем age. Используйте текущий год как константу при расчетах. Выполните скрипт. Откройте редактор и наберите скрипт.

```
oedit age
```

Скрипт:

```
echo Please enter your name:
read name
echo Please enter the year of your birth:
read year
let age=2006-year
echo Hello, $name. You are $age years old
```

Напишите скрипт dirlist, который выполняет следующие действия: запрашивает директорию для вывода; если пользователь ничего не вводит, то выводится домашняя директория пользователя.

Скрипт:

```
echo Directory:
read dir
if test -z "$dir"
then dir=$HOME
fi
ls -l $dir
```

Напишите скрипт dirlist2, который выполняет следующие действия: Запрашивает имя директории до тех пор, пока оно не будет введено. После этого выводит содержимое данной директории.

Скрипт:

```
while test "$1" = ""
do
echo Directory:
read dir
set "$dir"
done
ls -l $1
```

2.4 .Создание awk программ

Перед выполнением данной части лабораторной работы должны быть созданы следующие текстовые файлы sport и cars:

```
Jim  Fost  student     11.10.87   5   football
Mike Brown student     12.09.86   6   football
John Rodd  student     05.09.86   3   swimming
Bryan Talbot lecturer  04.07.75   10  basketball
Nick Shaw  student     03.08.87   5   swimming
Ann  Brook lecturer    08.04.79   6   swimming
Jane Fell  student     14.05.83   2   volley-ball
```

Рисунок 9 – Содержимое файла sport.

```
BMW : Paul : Brown : 12389
BMW : Jane : Tod :  34567
Chrysler : Mike : Bell : 67890
Chevrolet : Ben : Walsh : 45634
```

Рисунок 10 – Содержимое файла cars.

Для этого введите следующие команды:

```
touch sport
echo > Jim   Fost  student     11.10.87    5    football
echo > Mike  Brown student     12.09.86    6    football
echo > John  Rodd  student     05.09.86    3    swimming
echo > Bryan Talbot lecturer   04.07.75    10   basketball
echo > Nick  Shaw   student    03.08.87    5    swimming
echo > Nick  Shaw   student    03.08.87    5    swimming
echo > Ann   Brook  lecturer   08.04.79    6    swimming
echo > Jane  Fell   student    14.05.83    2    volley-ball
```

Аналогично создайте файл cars.

Введите awk команду, которая выведет первое поле каждой записи файла sport.

```
awk ' {print $1}' sport
```

Введите awk команду, которая выводит все поля для всех записей, начинающихся с имени “Jim”.

```
awk '$1 == "Jim" {print}' sport
```

Введите awk команду, которая для выводит первое и последнее поле для всех записей, начинающихся с имени “Jim”.

```
awk '$1 == "Jim" {print $1,$6}' sport
```

Запустите ISPF редактор при помощи команды oedit и создайте файл awk1. В файле awk1 напишите программу, которая выполняет следующие действия: для всех записей, начинающихся с “Jim”, выводит первое и последнее поле, и для всех записей, начинающихся с “Ann”, выводит второе и третье поле. В качестве входного файла укажите sport. Запустите программу.

Программа:

```
$1 == "Jim"  {print $1,$6}
$1 == "Ann" {print $2,$3}
```

Запуск на исполнение:

```
awk -f awk1 sport
```

Напишите awk программу awk2 , которая выводит название каждого поля файла sport. Названия полей: "Name Surname Occupation Date of Birth Exper Sport". Запустите вашу программу.

Программа:

```
BEGIN {print " Name Surname Occupation Date of Birth Exper Sport"}
      {print}
```

Команда:

```
awk -f awk2 sport
```

Напишите awk программу awk3, которая выполняет следующие действия: подсчитывает количество человек, занимающихся футболом и плаванием и выводит результаты.

Программа:

```
BEGIN {footcount = _ swimcount = _}
$6 == "football" {footcount = footcount + 1}
$6 == "swimming" {swimcount = swimcount + 1}
END {print footcount " people enjoy jogging."
print swimcount " people enjoy bridge."}
```

Команда:

```
awk -f awk3 sport
```

Напишите программу, которая выполняет следующие действия: использует символ (:) в качестве разделителя полей входного файла и выводит первое и второе поля, разделенные символом (:), затем выводит третье и четвертое поля, разделенные символом (,). В качестве входного файла укажите cars.lst. Запустите программу.

Программа:

```
BEGIN {FS = ":"}
{print $1 FS $2 ", " $3 ", " $4}
```

Команда:

```
awk -f awk4 cars.lst
```

Напишите awk программу awk5, которая выполняет следующие действия: использует символ (:) в качестве разделителя полей входного файла, считывает все записи, содержащие заглавную букву C и выводит эти строки без последнего поля.

Программа:

```
BEGIN {FS = ":"}
/C/ {print $1 ":" $2 ":" $3}
```

Команда:

```
awk -f awk5 cars.lst
```

Напишите awk команду awk6, которая выполняет следующие действия:

- Использует символ (:) в качестве разделителя полей входного файла.
- Считывает все записи, содержащие заглавную букву C.
- Выводит эти записи.
- Подсчитывает сумму по последним полям выведенных строк.
- Выводит результат суммирования.

Программа:

```
BEGIN {FS = ":"
total = _}
/C/
{amount = substr($4,2)
total = total + amount
 print $_}
END {print "The total is $"total}
```

Команда:

```
awk -f awk6 cars.lst
```

Создайте новый подкаталог и назовите его bin. Скопируйте в него все созданные программы.

Команды:

```
mkdir bin;
mv awk* bin
```

2.5 .Создание C программ

Используя редактор oedit, наберите следующую C программу и сохраните ее как hello.c .

Программа:

```
#include <stdio.h>
main()
{
printf(“Hello World \n”);
}
```

Используя команду c89 откомпилируйте и скомпонуйте программу. Команда: c89 hello.c Компилятором будут созданы файлы hello.o и a.out. Исполняемый файл: a.out, биты разрешений для исполняемого файла установлены следующие: -rwxr-xr-x. Запустите программу.

```
a.out
```

3 .Контрольные вопросы

3.1 .Чем отличается рабочий каталог от домашнего каталога?

3.2 .Как называется родительский каталог домашнего каталога?

3.3 .Как называется корневой каталог?

3.4 .Можете ли вы создать два каталога одной командой?

3.5 .Какие разрешения установлены для новых файлов?

3.6 .Какой inode номер у файла?

3.7 .Как вывести содержимое файла так, чтобы его содержимое последовательно выводилось на экран?

3.8 .Как определить размер файла?

3.9 .Что делает команда touch?

3.10 .Является ли жесткая ссылка файлом?

3.11 .Кто создает файл .sh history?

3.12 .Каким образом используются переменные среды?

3.13 .Сколько ID процессов создается при выполнении конвейера команд?

3.14 .Какой файл хранит параметры интерфейса пользовательской консоли?

3.15 .Как узнать размер свободного пространства в смонтированной файловой системе?

3.16 .Что такое shell скрипт?

3.17 .Какой командой можно вывести произвольную текстовую строку?

3.18 .Что такое Awk и для чего он используется?

3.19 .Из каких частей состоит Awk программа?

3.20 .С какими типами файлов работает Awk программа?

3.21 .Будет ли Awk программа корректно работать с текстовыми файлами Windows?

3.22 .Как запустить Awk программу?