МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА им. И.М.ГУБКИНА

Г.А. Карапетов , Ф.М. Барс

UNIX. Основы работы в системе.

Москва-2002г.

МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА им. И.М.ГУБКИНА

Кафедра разведочной геофизики и компьютерных систем

Г.А. Карапетов, Ф.М. Барс

UNIX. Основы работы в системе.

Учебное пособие

для студентов специальности 080400 “Геофизические методы поисков и разведки месторождений полезных ископаемых” специализации “Разведочная геофизика ” и “Компьютерные системы и технологии” (индекс ГФ).

Москва-2002г.

УДК 550.83

Г.А. Карапетов, Ф.М. Барс. UNIX. Основы работы в системе. Учебное пособие к практическим занятиям по курсу “Вычислительная математика и программирование”. М., РГУ нефти и газа им. И.М. Губкина, 2002. - с. 29
.

Данное пособие не ставит своей целью заменить обширный набор руководств и справочников, посвященных работе пользователей в операционной системе UNIX. Более того, в нем практически не затрагиваются вопросы системного администрирования, которые требуют детального понимания работы системы и не касаются непосредственно рядового пользователя. Оно возникло в результате более чем пятилетнего опыта чтения курсов для студентов специальности разведочная геофизика, ставящих своей целью получение навыков программирования типовых геофизических задач в среде, в которой функционируют программные продукты современных геолого-геофизических систем. Такие системы работают на мощных рабочих станциях в среде UNIX и требуют от пользователя наличия определенных навыков работы. Пособие ориентировано на специфику работы в аппаратно-программной среде, имеющейся в вычислительном центре кафедры, и не затарагивает всего многообразия возможностей UNIX как системы .

Пособие предназначено для студентов специальности **080400** специализаций “Разведочная геофизика " и “Компьютерные системы и технологии”. Оно также может оказаться полезным для студентов геолого-геофизических специальностей и специальностей связанных с разработкой месторождений, предполагающих специализироваться в области создания постоянно действующих моделей месторождений.

Рецензент – кандидат технических наук, Варов Е.Б.

Введение.

Система UNIX имеет более чем тридцатилетнюю историю. Множество профессионалов вложили в ее разработку свой труд, и сегодня использование этой системы стало практически повсеместно. Уместно отметить, что возникновение фонда бесплатного математического обеспечения (OSF- Open Software Foundation) связано именно с этой системой. Мощным толчком для развития UNIX послужил проект GNU и разработка версий UNIX для персональных ЭВМ: Linux и FreeBSD.

За время своего развития система претерпела значительные изменения, стала более функциональной и мощной. В то же время, базовые идеи, положенные в основу системы, сохранились и продолжают удивлять своим изяществом и простотой, обеспечивая не только жизнеспособностью UNIX, но и его сильное влияние на другие операционные системы.

В настоящее время существуют несколько десятков различных операционных систем, объединенных под названием UNIX. Имеются два основных направления System V (AT&T) и BSD (Беркли), различающихся с точки зрения организации и администрирования, хотя взаимное влияние этих направлений порой затрудняет однозначную классификацию отдельных версий систем. Несмотря на многообразие версий, основой всего семейства UNIX является принципиально одинаковая архитектура и стандартизованный интерфейс, что позволяет без особого труда переносить программы и работать на других версиях.

Среди коммерческих версий можно выделить AIX (IBM), Digital Unix (DEC), HP-UX (Hewlett Packard), IRIX (SGI), Solaris (SUN Microsystems). Такие системы функционируют на мощных рабочих станциях, однако,

мощность персональных компьютеров позволяет использовать их как в качестве X-терминалов, так и в качестве самостоятельных систем (stand along).

Учитывая конкретную аппаратную конфигурацию вычислительных средств и набор промышленных лицензионных продуктов, которыми располагает кафедра, в качестве базовых операционных системам применяются Linux (для PC) и Solaris (для SUN). В качестве языка программирования изучается C с объектами прикладного программного интерфейса Motif и Open Look. Для работы используются программные продукты проекта GNU (Gnu's Not Unix), а также специализированные геофизические программные продукты компаний Schlumberger (IESX) и Paradigm Geophysical (Focus и GeoDepth).

Операционная система UNIX.

Операционная система UNIX является многопользовательской многозадачной системой. Она построена в виде компактного ядра, обеспечивающего базовую функциональность операционной системы, и множества модулей, выполняющих отдельные функции. Ядро непосредственно взаимодействует с аппаратурой ЭВМ, обеспечивая минимальный набор услуг всем остальным программным компонентам: управление процессами, распределение системных ресурсов, операции ввода/вывода и т.д. Такая структура обеспечивает гибкость при инсталляции и конфигурировании системы (рис.1).

С точки зрения пользователя в операционной системе UNIX существует два основных типа объектов: файлы и процессы. Все программы (в том числе ядро, модули) и данные хранятся в файлах, работа с периферийными устройствами осуществляется с помощью операций записи/чтения в специальные файлы, механизм защиты данных реализован через права доступа к файлам, более того, доступ к памяти осуществляется через единый интерфейс файловой системы.

В то же время функциональность системы определяется выполнением соответствующих процессов, количество которых в каждый момент времени зависит от набора функций, возложенных на систему и может динамически меняться в зависимости от запросов пользователей.

Файловая система.

В UNIX, как и во многих современных операционных системах, файлы организованы в виде древовидной иерархической структуры, называемой файловой системой (рис. 2). Корневой каталог (root), обозначаемый /,является отправной точкой файловой системы. Каждый промежуточный узел файловой системы является каталогом. Конечные вершины ветвей дерева являются либо пустыми каталогами, либо файлами. Каталог считается пустым, если он не содержит никаких других файлов, кроме ссылок на текущий и родительский каталоги. Любой файл имеет имя, определяющее его положение в древовидной структуре. Полное (абсолютное) имя файла состоит из собственно имени файла и пути (path), представляющего собой последовательность ветвей (каталогов), определяющих его положение на дереве, начиная с корневого каталога, например:

```
/usr/openwin/bin/textedit
```

Относительное путевое имя задается относительно текущего каталога, полное путевое имя который можно определить, используя команду

```
$pwd
/usr/openwin
```

при этом относительное имя того же файла будет выглядеть `bin/textedit.`

Как правило, файловая система расположена на различных внешних устройствах и даже ЭВМ, но для рядового пользователя она представляет собой единое целое. Конкретная ее конфигурация определяется системным администратором и может динамически меняться. Имя файла является атрибутом файловой системы и не отражает содержимого файла. Оно содержится в каталогах в виде элемента каталога. Имя может состоять из

собственно имени и расширения (суффикса, возможно даже пустого) отделяемого точкой. Имеются определенные соглашения относительно суффиксов, в зависимости от назначения файла, однако их употребление является желательным, но не обязательным. Собственно расположение файла на внешнем устройстве определяется индексным дескриптором (inode). Связь между именем (в каталоге) и индексным дескриптором полностью характеризуют файл. Различают следующие типы файлов:

- обычный файл (regular), представляющий текстовую или двоичную информацию,
- каталог (directory), файл, содержащий имена находящихся в нем файлов и указатели на соответствующие индексные дескрипторы,
- специальный файл (device), обеспечивающий доступ к внешнему устройству,
- связь (link), специальный тип элемента каталога, позволяющий одному физическому файлу иметь несколько имен,
- именованный канал (FIFO), файл, предназначенный для связи между различными процессами,
- сокет (socket), коммуникационный порт, предназначенный для связи между различными процессами.

Поскольку UNIX является многопользовательской системой, в нем предусмотрен мощный механизм защиты данных от несанкционированного доступа, который определяется правами доступа (permissions). Имеется пользователь, обладающий неограниченными полномочиями – администратор системы. Для остальных пользователей существуют три основных класса доступа: для владельца файла (user), членов группы (group) и прочих (other) пользователей. Под группой понимаются пользователи, совместно работающие над какой-либо задачей или проектом.

Для каждого из них предусмотрены три типа прав: на чтение (read), на запись (write) и на выполнение (execute).

Список прав доступа можно получить с помощью команды

$ls -l

```
drwx------   34 user1   staff     4096 feb 21 18:23 .
drwxr-xr-x    8 nobody  staff     4096 feb  6  2001 ..
drwxr-xr-x    2 user1   staff     4096 jan 26  2000 include
drwxr-xr-x    2 user1   staff     4096 mar 23  2000 lib
-rw-r--r--    1 user1   staff     2016 mar 23 18:23 lsal.txt
lrwxrwxrwx    1 user1   staff     4096 feb 25  2000 libfa
-rwxr-xr-x    1 user1   staff     6345 feb 21 18:23 furie.c
```

Права доступа могут быть изменены либо его владельцем, либо администратором системы (root). Для изменения прав доступа используется команда **chmod**, имеющая следующий формат

chmod [ugoa][+-=][rwx] file

где

u	владелец-пользователь,
g	пользователи группы,
o	прочие пользователи,
a	все пользователи,
+	добавить,
-	удалить,
=	присвоить,
rwx	соответственно право на чтение, запись и выполнение.

Например,

$chmod a+w furie.c

позволяет разрешить запись в файл furie.c всем пользователям.

$chmod o-w+r,ug=w furie.c

позволяет удалить разрешение на запись и добавить разрешение на чтение для прочих пользователей и установить разрешение на запись для собственника и группы.

Возможен вариант установки прав доступа непосредственно в виде восьмеричного кода, например

$chmod 754 furie.c

означающий предоставление всех прав собственнику, на чтение и исполнение для группы и на чтение для всех прочих.

Необходимо знать, что установка некоторых ненужных прав (например, на исполнение для обычного текстового файла) может приводить в некоторых случаях к проблемам в функционировании некоторых компонентов системы, использующих их для действий по умолчанию. Для владельца данных всегда возможна переустановка прав доступа, вне зависимости от их конфигурации в данный момент, что может быть использовано для защиты собственных данных от случайного удаления.

Заметим, что для каталогов (обозначаемых в листинге d в первой позиции) значения прав доступа несколько отличаются. Символ **x** означает возможность поиска в каталоге (например, распечатка содержимого каталога), в то время как его отсутствие делает каталог “темным”, т.е. недоступным для просмотра. Право на запись следует давать чрезвычайно осмотрительно, поскольку в таком случае возможно удаление любых файлов даже при отсутствии разрешения на запись.

Имеются еще ряд дополнительных атрибутов, которые влияют на выполнение различных операций, но они, как правило, не касаются рядовых пользователей.

Команды работы с файлами и каталогами

Для создания файла с заданным именем может быть использована любая программа создающая какие-либо выходные данные. В частности, для копирования файла предназначена команда **cp** (copy)

$cp исходный файл целевой файл

В качестве целевого файла может быть и каталог. В этом случае файл с тем же именем копируется в целевой каталог.

Для переименования файла используется команда **mv** (move)., имеющая вид

$mv исходный файл целевой файл,

которая работает быстрее операции копирования, поскольку лишь меняет элемент каталога.

Для удаления файла используется команда **rm** (remove)

$rm целевой файл

при этом следует отметить, что обратное восстановление файла невозможно никакими средствами.

Для создания каталога предназначена команда **mkdir** (make directory)

$mkdir имя каталога

При этом, каталог может быть создан лишь внутри другого, если в него разрешена запись данному пользователю. Для удаления пустого каталога используется команда **rmdir** (remove directory)

$rmdir имя каталога

Если каталог не пуст, то выдается сообщение об ошибке.

Такую операцию для непустого каталога можно выполнить с помощью команды **rm** с флагом рекурсии

$rm –r целевой каталог

следует отметить, что в данном случае удаляется вся часть дерева файловой системы, содержащаяся в данном каталоге. При использовании данной команды следует проявлять большую осторожность и осмотрительность, чтобы не потерять навсегда результаты многодневной работы, поскольку в UNIX восстановление данных является принципиально невозможным, даже для администратора системы (здесь не имеется в виду резервное копирование – backup).

Аналогично, операцию копирования содержимого каталога вместе с вложенными подкаталогами можно провести, используя команду **cp** с флагом **–r**

$cp исходный каталог целевой каталог,

при этом исходный каталог должен быть открыт для чтения и просмотра, а целевой каталог для записи.

Процессы.

Понятие процесса в операционной системе UNIX играет ключевую роль. Любая программа, запущенная в операционной системе, имеет свою среду выполнения, т.е. различные системные ресурсы: услуги ядра операционной системы, ресурсы памяти, возможность доступа к внешним устройствам, стек и т.д. Таким образом, сама исполняемая программа в среде выполнения называется процессом. Однако не следует отождествлять программу и процесс, поскольку одна программа во время исполнения может порождать несколько процессов. В системе одновременно функционируют несколько процессов, которые можно просмотреть с помощью команды

$ps –e

```
PID  TTY  TIME CMD
  0    -  8:30 swapper
  1    -  8:30 init
.....................
372 pts/1 8:32 xterm
375 pts/1 8:32 mwm
378 pts/1 8:32 furie
402 pts/1 8:32 ps
```

Каждый процесс при запуске получает свой порядковый номер, который может быть использован для управления его состоянием. Операционная система UNIX обеспечивает иллюзию одновременного выполнения всех активных процессов, эффективно распределяя аппаратные ресурсы в соответствии с приоритетом процессов. Во время существования (жизненного цикла) процесс может находиться в одном из нескольких состояний, переходя

из одного в другое в зависимости от событий, происходящих в системе. Такими состояниями процесса могут быть:

- создан (с помощью fork), но еще не готов к запуску,
- не выполняется, но готов к запуску (runnable),
- выполняется в режиме задачи (user), т.е. обычные инструкции программы,
- выполняется в режиме ядра (kernel), т.е. инструкции операционной системы,
- находится в состоянии ожидания (sleep) из-за недоступного в данный момент ресурса или из-за переключения контекста на процесс большего приоритета,
- находится в состоянии зомби (defunct), когда процесс закончил свое существование (exit), но еще имеются следы его существования в системе.

Программа в процессе выполнения может “зависнуть”, т.е. прекратить реагировать на внешние воздействия или войти в бесконечный цикл. Это достаточно типичная ситуация, особенно на этапе разработке программы. В такой ситуации необходимо с помощью команды **ps** определить номер соответствующего процесса и исполнить команду **kill**:

$kill –9 378.

Любой процесс может быть запущен в фоновом режиме, если в конце команды использовать символ **&** (амперсенд). В этом случае управление вновь без ожидания завершения возвращается родительскому процессу. Всеми переключениями режимов процессов ведает планировщик, который выбирает процесс для запуска из очереди на основе критериев приоритетности.

Пользователи системы

Прежде чем начать работу в системе, пользователь должен получить свой бюджет (account) в ней. Иными словами, стать зарегистрированным

объектом, обладающим определенными правами и, как правило, имеющим свой (домашний) каталог. Каждый пользователь имеет свое уникальное регистрационное имя (login name) и пароль для входа в систему. Вся информация о пользователях хранится в системных файлах

/etc/passwd,
/etc/shadow ,
/etc/group,

к которым ограничен доступ. Создание нового пользователя и наделение его полномочиями является прерогативой администратора системы.

Пароль пользователя хранится в зашифрованном виде в файле /etc/shadow и недоступен для обычных пользователей. Однако каждый пользователь может изменить свой пароль с помощью команды

```
$passwd
Old password:
New password:
Retype new password:
```

При вводе пароль не отображается на экране, поэтому для контроля при изменении он повторяется дважды. Следует помнить, что паролем не должно быть словарное слово, фамилия или имя, номер телефона , адрес и другие легко вычисляемые комбинации. В то же время, не следует в качестве пароля выбирать трудно запоминаемые последовательности символов.

Среда пользователя

В функциональном отношении среда пользователя обладает всеми возможностями современной операционной системы: графическим многооконным интерфейсом, функциональными настраиваемыми меню,

возможностями навигации по файловой системе с помощью менеджера, технологиями drag-and-drop, различными оконными менеджерами и т.д. Однако следует отметить, что на рабочем столе обязательно присутствует окно, называемое терминальным, в котором присутствует базовая пользовательская среда и можно вводить и исполнять команды. Для интерпретации этих команд используется одна из командных оболочек (shell), которая является, по существу, командным интерпретатором. Собственно говоря, это первая программа, с которой приходится сталкиваться пользователю после окончания аутентификации.

К важнейшим оболочкам относятся оболочки приведенные в таб. 1.

Таблица 1.

Тип оболочки	Исполняемые сценарии
Баурна (Bourne Shell),	.profile
C (C Shell)	.login и .cshrc
Корна (Korne Shell)	.profile и .kshrc
BASH (Bourne Again Shell).	.bash_profile и .bashrc

Язык, на котором происходит общение с интерпретатором, является удобным средством программирования. Естественно, что он несколько отличается для различных интерпретаторов. Программа на языке оболочки называется сценарием (script). Сценарий представляет собой обычным текстовый файл, в котором описана последовательность действий для интерпретатора.

При входе в систему, в зависимости от типа оболочки исполняются некоторые стандартные сценарии инициализации, которые настраивают рабочий стол и окружение для конкретного пользователя. Те сценарии, которые расположены в домашнем каталоге пользователя, могут изменяться в зависимости от его потребностей и привычек. Следующий сценарий является типичным при работе пользователя в оболочке BASH.

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/bin:.:/usr/local/Acrobat4/bin:/usr/open
win/bin:/usr/X11R6/lib/X11/xfm:/usr/openwin/bin
BASH_ENV=$HOME/.bashrc
USERNAME="id -un"

if [ -f /tmp/.X0-lock ]
 then echo ""
 else xinit
fi
#
```

Строки, начинающиеся с символа # считаются комментариями и не интерпретируются. Данный сценарий является инициализационным. В нем вначале проверяется наличие файла с ресурсами **.bashrc** в домашнем каталоге пользователя и при его наличии его запуск. Затем определяются переменные окружения PATH, BASH_ENV и USERNAME, после чего запускается графический интерфейс. Заметим, что использование **xinit** вместо **startx** требует наличия ресурсного файла **.xinitrc**, в котором определены параметры графического интерфейса, в домашнем каталоге пользователя.

Общие свойства оболочек.

Каждая из оболочек UNIX имеет свои особенности. Стандартной для системы является оболочка Баурна, однако, более поздняя разработка BASH

обладает существенными преимуществами перед остальными. Возможно использование любой из оболочек, а при необходимости легко осуществить переход к другой оболочке.

В оболочка интерпретирует входную последовательность символов в команды операционной системы. Имеется ряд специальных символов, которые обозначают:

символ	значение
•	любую последовательность символов, включая пустую.
?	любой (но один) из допустимых в языке символов
[]	один из символов, содержащихся в скобках
\	знак маскировки интерпретации следующего символа.

Если команда помещена в обратные кавычки, то она исполняется, прежде чем осуществляется ее подстановка в качестве аргумента. Например,

$echo Today is `date`

```
Today is Fri Mar 14 12:15:38 2002
```

При запуске любой программы стандартно открываются три канала ввода-вывода с номерами (дескрипторами)

номер	назначение
0	стандартный ввод, который привязывается к специальному файлу клавиатуры,
1	стандартный вывод, который привязывается к терминальному окну, из которого запущена программа
2	стандартный вывод сообщений об ошибках, который привязывается также к терминальному окну.

Перенаправление стандартных каналов осуществляется с помощью символов

символ	назначение
>	стандартный вывод перенаправляется в файл, имя которого указывается следом за символом
<	стандартный ввод перенаправляется на файл, имя которого указывается следом за символом
>>	стандартный вывод добавляется к содержимому файла, имя которого указывается следом за символом. В случае отсутствия файла с данным именем аналогичен по действию символу >
>&	используется для перенаправления через дескрипторы, например, 2>&1.

Следующие примеры иллюстрируют возможные перенаправления:

$ls –al>catalogue.txt

$sort –nr < input_file >output_file 2>&1

Одной из эффектных возможностей является организация конвейеров, когда одна программа передает выходные данные непосредственно на вход другой программы. Для этого используется символ “|”, например:

$ls –al | grep pattern

С помощью конвейера можно реализовывать сложные составные команды из мелких команд и фильтров. Такой подход является основополагающим в UNIX при реализации задач управления.

Специальными символами обозначаются и некоторые каталоги:

~	домашний каталог (начальный каталог пользователя)
.	текущий каталог

..	родительский каталог (внешний по отношению к текущему)

X Windows

Система X Windows является графической оболочкой системы UNIX. Существующая отдельно от ядра системы, X Windows обладает высокой гибкостью в конфигурировании и не привязана к аппаратуре или к конкретной программе менеджеру окон. Система X Windows базируется на программной архитектуре клиент-сервер, в рамках которой программные приложения (клиенты) обращаются к серверу с запросами на выполнение операций, связанных с выводом изображения на экран. Сервер может обслуживать несколько клиентов, а один клиент может одновременно иметь связь с несколькими серверами. Поскольку связь между клиентом и сервером осуществляется через NFS, то естественно, что они могут находиться на разных ЭВМ. Учитывая, что протокол, лежащий в основе функционирования X Windows, является аппаратно-независимым, очевидно, что сеанс связи между клиентом и сервером должен проходить одинаково на различных аппаратных платформах и в различных модификациях UNIX.

Для идентификации целевого сервера служит переменная окружения DISPLAY, которая состоит из двух частей разделенных двоеточием. Первая часть определяет компьютер аналогично его определению в файле /etc/hosts, в то время как вторая-номер графического контроллера дисплея. Например:

DISPLAY=192.168.1.5:0 или
DISPLAY=linux5:0.0.

Для запуска X Windows имеется команда **xinit**, использование которой позволяет определить необходимое количество приложений-клиентов и ряд

других свойств. По умолчанию xinit использует дисплей :0, однако возможно его изменение из командной строки или в процессе работы.

Команда **xinit** использует файл конфигурации **.xinitrc**. В нем, помимо прочих операций конфигурирования, перечисляются программы, запуск которых пользователь считает необходимым в начале каждого сеанса работы с X Windows. К этим программам относится оконный менеджер и другие полезные приложения.

```
#!/bin/sh

sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

# start some nice programs
xsetroot -solid SteelBlue &
xclock -d -g -0+0 &
xterm -fn -cronyx-fixed-medium-r-semicondensed--13-120-
75-75-c-60-koi8-r&
mwm
```

Приведенный файл **.xinirc** является типичным при работе в Linux. По существу, он представляет собой сценарий для оболочки и является одним из первых приложений-клиентов сервера. После завершения его выполнение прерывается в обычном порядке, поэтому запуск всех команд, кроме завершающей, осуществляется в фоновом режиме. Исполнение последней команды в аналогичном режиме привело бы к немедленному завершению работы сервера сразу же после инициации, и, как следствие, к завершению работы всех приложений-клиентов.

В X Windows для описания атрибутов графического интерфейса используются ресурсы. Значения ресурсов зависят от их природы. В качестве ресурсов могут выступать цвет фона, толщина и цвет окантовки окон, тип и размер шрифта и т.д. Всякий ресурс имеет свой идентификатор, в котором отражен иерархический подход к построению имен ресурсов. Пользовательские установки описываются в файле **.Xdefaults**, расположенном в домашнем каталоге пользователя. Они автоматически загружаются при инициализации графического сервера. Настройки ресурсов производятся путем редактирования содержимого файла **.Xdefaults** и последующего использования команды **xrdb**.

Наиболее употребительные приложения-клиенты.

К числу наиболее употребительных приложений доступных обычному пользователю системы X Windows относятся программные продукты, которые принято использовать для организации рабочего стола (desktop). Согласно основной концепции UNIX состав и количество прикладных компонентов зависят от потребностей конкретного пользователя и могут меняться в широких пределах. Детальные настройки всех компонентов описаны в ON-line руководствах и требуют от пользователя достаточно серьезного понимания процесса, поэтому в данном разделе будут перечислены наиболее употребительные приложения и некоторые аспекты их первоначальных настроек, позволяющих начать их использование.

Xterm – графическое терминальное окно, позволяющее обеспечить выполнение всех команд операционной системы UNIX. Именно с ним в UNIX связаны понятия управляющего терминала в графическом режиме и стандартного устройства ввода/вывода (набор на клавиатуре отображается именно на терминале). Такое окно, несмотря на мощный графический интерфейс, является традиционным для рабочего стола пользователя и позволяет и позволяет наиболее эффективно выполнять сложные операции управления работой. Особенно эффективно использование графической

консоли при работе в сложном сеансе, когда используется удаленный доступ с одновременной работой на нескольких ЭВМ

Xman – броузер справочного руководства пользователя, точнее его графическая оболочка Является одной их наиболее употребительных и полезных утилит UNIX, поскольку позволяет обеспечить форматирование текста со справочной информацией в отдельном окне. Для получения информации по интересующему вопросу, ее заголовок выбирается из панели Options с помощью мыши.

Xfm – файловый менеджер системы X Windows. В его окне отображаются каталоги и файлы файловой системы в той конфигурации, которая существует в данный момент (т.е. всех подмонтированных файловых систем вне зависимости от типа). Настройка файлового менеджера осуществляется в домашнем каталоге пользователя в подкаталоге .xfm. Для его создания необходимо при первоначальном сеансе запустить сценарий **xfm.install**. ОТ эффективности настройки **xfm** в значительной мере зависит удобство работы в графическом режиме. Для указания действий по умолчанию связанных с характером соответствующего файла (определяется типом по некоторой совокупности характеристик) необходимо отредактировать файл ~/.xfm/xfm-apps. В нем определяется имя запускаемой программы, например, редактора для текстового файла и т.д. Характер графического отображения информации в виде пиктограмм задается в файле ~/.xfm/xfmrc.

Xclock – одно из многочисленных приложений, позволяющих индицировать на экране время и дату. По умолчанию запускается в виде циферблата со стрелками, при задании флага –d имеет вид цифровых часов.

Textedit – текстовый редактор, традиционный для OpenLook (SUN). Используется, поскольку, в дальнейшем предусматривается работа в Solaris, и пользователю желательно получить определенные навыки по работе с ним. Гамма текстовых редакторов чрезвычайно широка: emaxs (GNU), xedit, joe,

jed, gedit и т.д. Их использование зависит от потребностей и привычек пользователя и от конкретной конфигурации.

Для пользователей, привыкших к интерфейсу типа Norton Commander, имеется приложение **mc (M**idnight **C**ommander), которое по внешнему виду практически не отличается от него, хотя значительно более функционально.

Учитывая то обстоятельство, что вызов из командной строки с учетом возможных настроек может быть достаточно длинным, целесообразно использовать вышеупомянутые приложения через меню менеджера окон, например, mwm:

```
!!
!!  DEFAULT Mwm RESOURCE DESCRIPTION FILE
!!
!!       $HOME/.mwmrc
Menu DefaultRootMenu
{
    no_label                f.separator
    "Motif Root Menu"    f.title
    no_label                f.separator
    "New Window"     f.exec "xterm -ls -fn  -cronyx-
fixed-medium-r-semicondensed--13-120-75-75-c-60-koi8-r&"
    "Programs..."        f.menu Programs
    "Utilities..."       f.menu Utilities
    "Hosts..."           f.menu Hosts
    "Acroread"           f.exec
"/home/public/Acrobat4/bin/acroread&"
    "Restart mwm"        f.restart
    "Quit..."            f.quit_mwm
}


Menu Programs
{
    "XTerm"          f.exec "xterm -fn  -cronyx-fixed-
medium-r-semicondensed--13-120-75-75-c-60-koi8-r&"
    "File manager"       f.exec "xfm &"
    "GMC"                f.exec "gmc &"
    "Midnight Commander"  f.exec "xterm -fn -cronyx-
fixed-medium-r-semicondensed--13-120-75-75-c-60-koi8-r -
title \"mc\" -e mc -c&"
```

```
    "Editor"          f.exec "/usr/openwin/bin/textedit
&"
    "Mail"            f.exec "xmail -m &"
    "Calculator"      f.exec "xcalc &"
    "Manual"          f.exec "xman &"
    "Xclock"          f.exec "xclock &"
    "Clipboard"           f.exec "xclipboard &"
    "Magnify glass"       f.exec "xmag &"
}
```

которое определяется в файле настроек .mwmrc в домашней директории или через укороченный вызов, реализуемый с помощью оболочки, например, с помощью директивы alias.

Утилита make

Утилита **make** предназначена для упрощения обслуживания таких объектов как исходные, объектные и исполнимые файлы. Она чрезвычайно полезна при разработке сложных программ, библиотек и других компонентов математического обеспечения, состоящих из множества модулей, содержащихся в различных файлах. Утилита в соответствии со структурой задания, которое, как правило, помещается в файл с именем makefile, отслеживает зависимости между модулями и в случае изменения модулей автоматически повторно компилирует их, исключая ненужные действия с модулями, код которых не изменяется. Это позволяет избежать выполнения огромного количества непроизводительной работы в терминальном окне.

Совместно с системой SCCS – Source Code Control System (системой управления исходными кодами), позволяющей отслеживать различные версии исходных кодов, представляет собой эффективное средство управления разработкой сложных проектов.

Понятно, что утилита make имеет очень широкие функции, поскольку в ее задание могут быть включены различные командные строки, правила, макроопределения и статические и динамические макроподстановки, проверки условий и многое другое. Детальное описание синтаксиса можно

найти в соответствующих руководствах и ON-LINE документации. Простейший makefile содержит единственную цель в следующем формате:

Цель: [зависимости]

[команды]

Целевое имя заканчивается **:** . Строка завершается списком зависимости. Последующие строки являются командными линиями.

Рассмотрим одну из типичных структур makefile, предназначенную для компиляции и сборки программы `Signal_analysis.`

```
#  makefile for Signal_analysis Program
     CC = gcc
     FLAGS = -g
     PROGR = Signal_analysis

     incl = -I/home/STAFF/user/include
     libs = -L/home/STAFF/user/lib -lPlotW -lNUtil -L/usr/lib -lm
-L/usr/X11R6/lib -lXm -lXpm -lXext -lXt -lX11

     SRCS = menu_bar.c fftcwp.c File_operation.c read_data.c
write_data.c DFT_dir.c Signal.c Ph_corr.c spectr.c mph_tr.c
DFTr_inv.c DFTi_inv.c Zero_ph.c
     OBJS = $(SRCS:.c=.o)
#-----------------------------------------------------
$(PROGR): $(OBJS)
     $(CC) -o $(PROGR) $(FLAGS) $(OBJS) $(libs)
#-----------------------------------------------------
.c.o:
     $(CC) -c $(FLAGS) $< $(incl)
#-----------------------------------------------------
remove:
     rm -f *.o core
     rm -f $(PROGR)
```

В начале файла производится определение некоторых переменных, которые в дальнейшем участвуют в макроподстановках в форме **$(value).** Далее определяются дополнительные пути **incl** и **libs** для поиска файлов-заголовков (header files) и библиотек объектных модулей. Определяются исходные модули, используемые для создания программы (**SRCS**), и явное правило преобразования исходных модулей в объектные (**OBJS**). Затем следует основная цель **$(PROGR)**, в которой определено правило сборки исполнимой программы, и дополнительные цели **.c.o:**, определяющая процесс преобразования исходных файлов на языке С в объектные, и **remove:**, предназначенная для удаления всех вновь образованных файлов, включая и **core**-файл, образующийся при неправильном завершении основной программы.

В задании используется стандартный динамический макрос **$<** , обозначающий имена связанных файлов, выбираемых на основе неявных правил. Другими словами, подставляются имена измененных после предыдущей сборки исходных файлов из набора **SRCS**.

Следует отметить, что при вызове утилиты **make** все исходные файлы должны находиться в текущей директории или быть определены явно с помощью путевого имени. Первое предпочтительнее, поскольку упрощает задание и предотвращает “расплывание” связанных программных компонентов по файловой системе. Такая практика позволяет более эффективно управлять проектом. Это не касается библиотек и файлов-заголовков к ним, которые в соответствие с системными соглашениями желательно помещать в домашнем каталоге в **~/lib** и **~/include** соответственно. В тех случаях, когда необходимо исполнить лишь часть задания (отдельную цель), то ее нужно явно указывать утилите в командной строке при вызове. При наличии makefile с другим именем необходимо использовать флажок -f, например:

$make –f имя_задания или

$make –f цель или

$make –f имя_задания цель.

Список литературы.

1. М. Банахан, Э.Раттер. Введение в операционную систему UNIX., Радио и связь,
1986г.
2. Р. Готье. Руководство по операционной системе UNIX. М., Финансы и
статистика, 1985г.
3. Билл Болл, Девид Питс и др. Red Hat Linux 7. Энциклопедия. М.,С.-П.,Киев,
Диал Софт, 2001г., 592 стр. с ил.
4. Девид Белден, Роберт Неймер. UNIX. Руководство по операционной системе.
Изд. 6-ое. М., С.-П., Киев, Wiliams, 2002г., 789 стр. с ил.
5. Джеймс С. Армстронг (мл.). Секреты UNIX. 2-ое изд. Киев, Диалектика, 2001г.,
576 стр. с ил.
6. Дунаев С. UNIX.System V. Release 4.2. Общее руководство. М., Диалог МИФИ,
1997г., 287 стр. с ил.
7. Б. Керниган, Р. Пайк. UNIX – универсальная среда программирования. М.,
Финансы и статистика, 1992г., 327с. с .ил.
8. Э. Немет. UNIX: руководство системного администратора. Киев, BHV, 1997г.,
2-ое. изд., 892 стр. с ил.
9. Р. Петерсен. Linux: руководство по операционной системе. Киев, BHV, 1997г.,
2-ое.изд., 688 стр. с. ил.
10. Робачевский А. Операционная система UNIX. С.-П., BHV, 1997г., 528 стр.
с ил.
11. Стен Келли-Бутл. Введение в UNIX. М., Лори,1995г., 340 стр. с ил.
12. Теренс Чанг. Системное программирование на C++ для UNIX. Киев, BHV,
1997г., 592 стр. с ил.

Оглавление.

Стр.

Карапетов Григорий Артаваздович, Барс Фания Мансуровна

UNIX. Основы работы в системе.

Св. тематический план 2002 г.

Подписано в печать
Объем 1.7уч.-изд. л.
Заказ

Тираж 50 экз.
Формат 60 x 90 / 16

117917, Ленинский просп., 65, РГУ нефти и газа им.И.М.Губкина.
Отдел оперативной полиграфии