Семинар 1. Основы ОС UNIX

Основной постоянно функционирующей частью операционной системы UNIX является ее ядро. Другие программы (будь то системные программы или пользовательские) могут общаться с ядром посредством системных вызовов, которые по сути дела являются прямыми точками входа программ в ядро. При исполнении системного вызова программа пользователя временно переходит в привилегированный режим, получая доступ к данным или устройствам, которые не доступны при работе в режиме пользователя.

Реальные машинные команды, которые требуются для активизации системных вызовов, естественно, отличаются от машины к машине, наряду со способом передачи параметров и результатов между вызывающей программой и ядром. Однако с точки зрения программиста на языке С использование системных вызовов ничем внешне не отличается от использования других функций стандартной ANSI библиотеки языка С, таких как, например, функции работы со строками *strlen()*, *strcpy()* и т.д. Стандартная библиотека в UNIX - *libc* - обеспечивает С интерфейс к каждому системному вызову. Это приводит к тому, что системный вызов выглядит как функция на языке С для программиста. Более того, многие из уже известных вам стандартных функций, например, функции для работы с файлами: *fopen(), fread(), fwrite()* при реализации в операционной системе UNIX при своей работе будут применять различные системные вызовы. По ходу нашего курса нам придется ознакомиться с большим количеством разнообразных системных вызовов и их С-интерфейсами.

Большинство системных вызовов, возвращающих целое значение, использует значение *-1* при возникновении ошибки и значение большее или равное 0 при нормальном завершении. Системные вызовы, возвращающие указатели, обычно для идентификации ошибочной ситуации пользуются значением *NULL*. Для точного определения причины ошибки С-интерфейс предоставляет глобальную переменную *errno*, описанную в файле *<errno.h>* вместе с ее возможными значениями и их краткими определениями. Заметим, что анализировать значение переменной *errno* необходимо сразу после возникновения ошибочной ситуации, так как успешно завершившиеся системные вызовы не изменяют ее значения. Для получения символьной информации об ошибке на стандартном выводе программы для ошибок (по умолчанию экран вашего терминала) может применяться стандартная UNIX функция *perror()*.

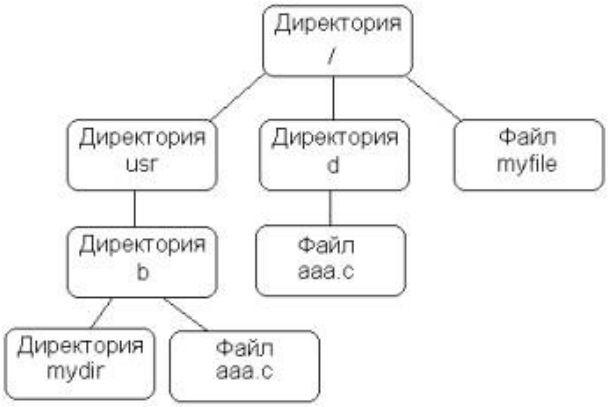
Понятия login'a & password'a. Операционная система UNIX является многопользовательской операционной системой. Для обеспечения безопасной работы

пользователей и целостности системы любой доступ к ней должен быть санкционирован. Для каждого пользователя, которому разрешен вход в систему, заводится специальное регистрационное имя - username или login и сохраняется специальный пароль - password, соответствующий этому имени.

Упрощенное понятие об устройстве файловой системы в UNIX. Полные и относительные имена файлов. В операционной системе UNIX существует три базовых понятия: "процесс", "файл" и "пользователь". С понятием "пользователь" вы только что уже столкнулись и будете сталкиваться в дальнейшем при изучении средств защиты информации в операционной системе UNIX. Понятие "процесс" характеризует динамическую сторону происходящего в вычислительной системе. Понятие "файл" характеризует статическую сторону вычислительной системы.

Из предыдущего опыта работы с вычислительной техникой вы уже имеете некоторое представление о файле, как о некотором именованном наборе данных, хранящегося где-нибудь на магнитных дисках или лентах. Для нашего сегоднящнего обсуждения нам достаточно такого понимания, чтобы упрощенно разобраться в том, как организована работа с файлами в операционной системе UNIX.

Все файлы, которые доступны операционной системе UNIX, как и в уже известных вам операционных системах, объединяются в древовидную логическую структуру. Файлы могут объединяться в каталоги или директории. Не существует файлов, которые не входили бы в состав какой-либо директории. Директории в свою очередь могут входить в состав других директорий. Допускается существование пустых директорий, в которые не входит ни один другой файл, и ни одна другая директория (см. рисунок). Среди всех директорий существует только одна директория, которая не входит в состав других директорий - ее принято называть корневой. На настоящем уровне нашего незнания UNIX мы можем заключить, что в файловой системе UNIX присутствует, по крайней мере, 2 типа файлов: обычные файлы, которые могут содержать тексты программ, исполняемый код, данные и т.д. - их принято называть регулярными файлами, и директории.



Каждому файлу (регулярному или директории) должно быть присвоено имя. В различных версиях операционной системы UNIX существуют различные ограничения на построение имени файла. В стандарте POSIX на интерфейс системных вызовов для операционной системы UNIX содержится лишь три явных ограничения:

1) Нельзя создавать имена большей длины, чем это предусмотрено операционной системой (для Linux - 255 символов).

2) Нельзя использовать символ *NUL* (не путать с указателем *NULL* !) - он же символ с нулевым кодом, он же признак конца строки в языке С.

3) Нельзя использовать символ '/'.

Также нежелательным является использование символов "звездочка" - '*', "знак вопроса" -'?', "кавычка " -'\"', "апостроф" - '\'', "пробел" - ' ' и "обратный слэш" -'\\' (символы записаны в нотации символьных констант языка С). Единственным исключением из перечисленных правил служит корневая директория, которая всегда имеет имя "/". Эта же директория по вполне понятным причинам является единственным файлом, обязанным иметь уникальное имя во всей файловой системе. Для всех остальных файлов имена должны быть уникальными только в рамках той директории, в которую они непосредственно входят. Каким же образом отличить два файла с именами "aaa.c", входящими в директории "b" и "c" на нашем рисунке, чтобы было понятно о каком из них идет речь? Здесь на помощь приходит понятие полного имени файла.

Давайте мысленно построим путь от корневой вершины нашего дерева файлов к интересующему нас файлу и выпишем все имена файлов (т.е. узлов дерева), встречающиеся на нашем пути, например, "/ usr b aaa.c". В этой последовательности первым будет всегда стоять имя корневой директории, а последним - имя интересующего нас файла. Отделим имена узлов друг от друга в этой записи не пробелами, а символами "/", за исключением имени корневой директории и следующего за ним имени ("/usr/b/aaa.c"). Полученная запись однозначно идентифицирует файл во всей логической конструкции файловой системы. Такая запись и получила название полного имени файла.

Понятие о текущей директории. Команда pwd. Относительные имена файлов. Полные имена файлов могут включать в себя достаточно много имен директорий и быть очень длинными, с ними не всегда удобно работать. Тогда на помощь нам могут прийти понятия текущей или рабочей директории и относительного имени файла.

Для каждой работающей программы в операционной системе, включая командный интерпретатор (*shell*) , который обрабатывает вводимые вами команды и высвечивает приглашение к их вводу, некоторая директория в логической структуре файловой системы

назначается текущей или рабочей для данной программы. Узнать, какая директория является текущей для вашего командного интерпретатора, можно с помощью команды операционной системы *pwd*.

Зная текущую директорию, мы можем проложить путь по графу файлов от текущей директории к интересующему нас файлу. Запишем последовательность узлов, которые встретятся на этом пути следующим образом. Узел, соответствующий текущей директории в запись не включаем. При движении по направлению к корневому каталогу каждый новый встретившийся узел будем обозначать двумя символами "точка" - "..", а при движении по направлению от корневого каталога будем записывать имя встретившегося узла. Разделим обозначения, относящиеся к разным узлам в этой записи символами "/". Полученную строку принято называть относительным именем файла. Относительные имена файлов меняются при смене рабочего каталога. Так, в нашем примере, если рабочий каталог - это директория "/d", то для файла "/usr/b/aaa.c" его относительное имя будет "../usr/b/aaa.c", а если рабочий каталог - это директория "/usr/b", то его относительное имя - "aaa.c".

Для полноты картины имя текущего каталога можно тоже вставлять в относительное имя файла, обозначая текущий каталог одиночным символом "точка" - ".". Тогда наши относительные имена будут выглядеть как "./../usr/b/aaa.c" и "./aaa.c" соответственно.

Программы, запущенные вами с помощью командного интерпретатора, будут иметь в качестве рабочей директории его рабочую директорию, если вы внутри этих программ не измените ее расположение с помощью специального системного вызова.

Домашняя директория пользователя и ее определение. Для каждого нового пользователя в системе заводится специальная директория, которая становится текущей сразу после входа в систему. Эта директория получила название домашней директории пользователя. Воспользуйтесь командой *pwd* для определения своей домашней директории.

Команда *man* - универсальный справочник. По ходу изучения операционной системы UNIX вам часто будет требоваться информация о том, что делает та или иная команда или системный вызов, какие у них параметры и опции, для чего предназначены некоторые системные файлы, и каков их формат и т.д. Мы постарались, по мере возможности, занести для большинства используемых в курсе команд и системных вызовов их описания в наш гипертекст, куда вы можете попасть по соответствующим

ссылкам (как, например, здесь для команды *pwd*), однако иногда для полной информации мы отсылаем вас к UNIX Manual - руководству по операционной системе UNIX. К счастью, большинство информации в UNIX Manual доступно для пользователей в интерактивном режиме с помощью утилиты *man*.

Пользоваться утилитой *man* достаточно просто - наберите команду

man имя,

где *имя* - это имя интересующей вас команды, утилиты, системного вызова, библиотечной функции или файла. Попробуйте с ее помощью посмотреть информацию о команде *pwd*.

Пролистать страницу полученного описания, если оно все не поместилось на экране, можно нажав клавишу <пробел>. Для прокрутки одной строки воспользуйтесь клавишей <Enter>. Вернуться на страницу назад позволит одновременное нажатие клавиш <Ctrl> и <b>. Выйти из режима просмотра информации можно нажатием клавиши <q>.

Иногда имена команд интерпретатора и системных вызовов или какие-либо еще имена совпадают. Тогда для правильного выбора интересующей вас информации необходимо задать утилите *man* категорию, к которой относится интересующая вас информация (номер раздела). Деление информации по категориям может слегка отличаться от одной версии UNIX к другой. В Linux, например, принято следующее разделение:

1) Исполняемые файлы или команды интерпретатора.
2) Системные вызовы.
3) Библиотечные функции.
4) Специальные файлы (обычно файлы устройств).
5) Формат системных файлов и принятые соглашения.
6) Игры (обычно отсутствуют).
7) Макропакеты и утилиты - такие как сам man.
8) Команды системного администратора.
9) Подпрограммы ядра (нестандартный раздел).

Если вы знаете раздел, к которому относится информация, то утилиту *man* можно вызвать в Linux с дополнительным параметром

man номер_раздела имя.

В других операционных системах этот вызов может выглядеть иначе. Для получения точной информации о разбиении на разделы, форме указания номера раздела и дополнительных возможностях утилиты *man* наберите команду

man man

Команды *cd* - смены текущей директории и *ls* - просмотра состава директории. Для смены текущей директории командного интерпретатора можно воспользоваться командой *cd* от (change directory). Для этого необходимо набрать команду в виде:

cd имя_директории,

где *имя_директории* - полное или относительное имя директории, которую вы хотите сделать текущей. Команда *cd* без параметров сделает текущей директорией вашу домашнюю директорию.

Просмотреть содержимое текущей или любой другой директории можно, воспользовавшись командой *ls* (от list). Если ввести ее без параметров, эта команда распечатает вам список файлов, находящихся в текущей директории. Если же в качестве параметра задать полное или относительное имя директории:

ls имя_директории,

она распечатает список файлов в указанной директории. Надо отметить, что в полученный список не войдут файлы, имена которых начинаются с символа "точка" - '.'. Такие файлы обычно создаются различными системными программами для своих целей (например, для настройки их работы). Посмотреть полный список файлов можно, дополнительно указав команде *ls* опцию *-a*, т.е. набрав ее в виде

ls -a

или

ls -a имя_директории.

У команды *ls* существует и много других опций, часть из которых мы рассмотрим на семинарах позже. Для получения полной информации о команде *ls* воспользуйтесь утилитой *man.*

Путешествие по директорной структуре. Пользуясь командами *cd, ls* и *pwd* попутешествуйте по директорной структуре вашей операционной системы и порассматривайте ее содержимое. Возможно, зайти в некоторые директории или посмотреть их содержимое вам не удастся. Это связано с работой механизма защиты файлов и директорий, о котором мы поговорим немного позже. Не забудьте в конце путешествия вернуться в вашу домашнюю директорию командой:

cd ~/

Команда *cat* и создание файла. Перенаправление ввода и вывода. Мы умеем перемещаться по логической структуре файловой системы и рассматривать ее содержимое. Хотелось бы уметь еще и просматривать содержимое файлов, и создавать их.

Для просмотра содержимого небольшого текстового файла на экране можно воспользоваться командой *cat.* Если набрать ее в виде

cat имя_файла,

то на экран выплеснется все его содержимое.

Внимание! Не пытайтесь рассматривать на экране содержимое директорий - все равно не получится! Не пытайтесь просматривать содержимое неизвестных файлов, особенно если вы не знаете, текстовый он или бинарный. Вывод на экран бинарного файла может привести к непредсказуемому поведению вашего терминала. Если даже ваш файл и текстовый, но большой, то все равно вы увидите только его последнюю страницу. Большой текстовый файл удобнее рассматривать с помощью утилиты *more* (обращайтесь к UNIX Manual для описания ее использования). Команда же *cat* будет интересна нам с другой точки зрения.

Если мы в качестве параметров для команды *cat* зададим не одно имя, а имена нескольких файлов

cat файл1 файл2 ... файлN,

то на экран последовательно выплеснется все их содержимое в указанном порядке. Вывод команды *cat* можно перенаправить с экрана терминала в какой-нибудь файл, воспользовавшись символом перенаправления выходного потока данных - знаком "больше"- ">". Команда

cat файл1 файл2 ... файлN > файл_результата

сольет содержимое всех файлов, чьи имена стоят перед знаком ">", воедино в *файл_результата* - конкатенирует их (от слова concatenate и произошло ее название). Прием перенаправления выходных данных со стандартного потока вывода (экрана) в файл является стандартным для всех команд, выполняемых командным интерпретатором. Вы можете получить файл, содержащий список всех файлов текущей директории, если выполните команду *ls -a* с перенаправлением выходных данных

ls -a > новый_файл.

Если имена входных файлов для команды *cat* не заданы, то она будет использовать в качестве входных данных информацию, которая будет вводиться с клавиатуры, до тех пор, пока вы не наберете признак окончания ввода - комбинацию клавиш <CTRL> и <d>.

Таким образом, команда

cat > новый_файл

позволяет создать новый текстовый файл с именем *новый_файл* и содержимым, которое пользователь введет с клавиатуры. У команды *cat* существует множество различных опций. Посмотреть ее полное описание можно в UNIX Manual.

Заметим, что наряду с перенаправлением выходных данных существует способ перенаправить входные данные. Если во время работы некоторой команды требуется ввод данных с клавиатуры, то можно положить их заранее в файл, а затем перенаправить стандартный ввод этой команды с помощью знака "меньше" - "<" и следующего за ним имени файла с входными данными. Другие варианты перенаправления потоков данных можно посмотреть в UNIX Manual для вашего командного интерпретатора.

Простейшие команды работы с файлами - *cp, rm, mkdir, mv.* Для нормальной работы с файлами необходимо не только уметь создавать файлы, просматривать их содержимое и перемещаться по логическому дереву файловой системы. Нам нужно уметь создавать свои собственные поддиректории, копировать и удалять файлы, переименовывать их. Это минимальный набор операций, без которого нельзя чувствовать себя комфортно.

Для создания новой поддиректории используется команда *mkdir* (сокращение от make directory). В простейшем виде команда выглядит следующим образом:

mkdir имя_директории,

где *имя_директории* - полное или относительное имя создаваемой директории. У команды *mkdir* имеется набор опций, описание которых можно посмотреть с помощью утилиты *man*.

Для копирования файлов может быть использована команда *cp* (сокращение от copy). Команда *cp* умеет копировать не только отдельный файл, но и набор файлов, и даже целиком директорию вместе со всеми входящими в нее поддиректориями (рекурсивное копирование). Для задания набора файлов могут использоваться шаблоны имен файлов. Точно также шаблон имени может быть использован и в командах переименования файлов и их удаления, которые мы рассмотрим ниже.

Для удаления файлов или директорий применяется команда *rm* (сокращение от remove). Если вы хотите удалить один или несколько регулярных файлов, то простейший вид команды *rm* выглядит следующим образом:

rm файл1 файл2 ... файлN,

где *файл1, файл2, ... файлN* - полные или относительные имена регулярных файлов, которые вы хотите удалить. Вместо непосредственно имен файлов могут использоваться их шаблоны. Если вы хотите удалить одну или несколько директорий вместе с их содержимым (рекурсивное удаление), то к команде добавляется опция *-r*:

rm -r дир1 дир2 ... дирN,

где *dup1, dup2, ... dupN* - полные или относительные имена директорий, которые вы хотите удалить. Вместо непосредственно имен директорий также могут использоваться их шаблоны. У команды *rm* есть еще набор полезных опций, которые описаны в UNIX Manual.

Командой удаления файлов и директорий следует пользоваться с осторожностью. Удаленную информацию восстановить невозможно. Если вы системный администратор и ваша текущая директория - это корневая директория, пожалуйста, не выполняйте команду "*rm -r* *"!

Для переименования файла или его перемещения в другой каталог применяется команда *mv* (сокращение от move). Для задания имен перемещаемых файлов в ней тоже можно использовать их шаблоны.

Система Midnight Commander - *mc.* Наверное, вы уже убедились в том, что работа в UNIX исключительно на уровне командного интерпретатора и встроенных редакторов далека от уже привычных для вас удобств. Но не все обстоит так плохо. Существует разнообразные пакеты, облегчающие жизнь пользователя в UNIX. К таким пакетам следует отнести Midnight Commander - аналог программ Norton Commander для DOS и FAR для Windows 9x и NT - со своим встроенным редактором, запускаемый командой *mc*. Информацию о них вы можете найти в UNIX Manual. Редактор так же можно вызвать командой *mcedit*.

Команда *ls* с опциями *-al*. Использование команд *chmod* и *umask*. Посмотреть подробную информацию о файлах в некоторой директории, включая имена хозяина, группы хозяев и права доступа, можно с помощью уже известной нам команды *ls* с опциями *-al*. В выдаче этой команды третья колонка слева содержит имена пользователей хозяев файлов, а четвертая колонка слева - имена групп хозяев файла. Самая левая колонка содержит типы файлов и права доступа к ним. Тип файла определяет первый символ в наборе символов. Если это символ '*d*' - то тип файла - директория, если там стоит символ '-', то это - регулярный файл. Следующие три символа определяют права доступа для хозяина файла, следующие три - для пользователей, входящих в группу хозяев файла, и последние три - для всех остальных пользователей. Наличие символа (*r, w* или *x*), соответствующего праву, для некоторой категории пользователей означает, что данная категория пользователей обладает этим правом.

Вызовите команду *ls -al* для своей домашней директории и проанализируйте ее выдачу. Хозяин файла может изменять права доступа к нему, пользуясь командой *chmod*.

Создайте новый файл и посмотрите на права доступа к нему, установленные системой при его создании. Чем руководствуется операционная система при выставлении этих прав? Для этого она использует маску создания файлов для программы, которая файл создает. Изначально для программы-оболочки она имеет некоторое значение по умолчанию. Изменить текущее значение маски для программы-оболочки или посмотреть его можно с помощью команды *umask.* Если вы хотите изменить его для Midnight Commander, необходимо выйти из *mc*, выполнить команду *umask* и запустить *mc* снова. Маска создания файлов не сохраняется между сеансами работы в системе. При новом входе в систему значение маски снова будет установлено по умолчанию.

Системные вызовы *getuid* и *getgid*. Узнать идентификатор пользователя, запустившего программу на исполнение, - UID и идентификатор группы, к которой он относится, - GID можно с помощью системных вызовов *getuid()* и *getgid(),* применив их внутри этой программы.

Компиляция программ на языке С в UNIX и запуск их на счет. Теперь мы практически созрели для того, чтобы написать первую программу в нашем курсе. Осталось только научиться компилировать программы на языке С и запускать их на счет. Для компиляции программ в Linux мы будем применять компилятор *gcc*. Для того, чтобы он нормально работал, необходимо, чтобы исходные файлы, содержащие текст программы, имели имена, заканчивающиеся на .c. В простейшем случае откомпилировать программу можно, запустив компилятор командой

gcc имя_исходного_файла.

Если программа была написана без ошибок, то компилятор создаст исполняемый файл с именем *a.out*. Изменить имя создаваемого исполняемого файла можно, задав его с помощью опции *-o* :

gcc имя_исходного_файла -o имя_исполняемого_файла.

Компилятор *gcc* имеет несколько сотен возможных опций. Получить информацию о них вы можете в UNIX Manual.