МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ
Кафедра вычислительных машин, комплексов, систем и сетей

Н.И. РОМАНЧЕВА

оглавление

ПОСОБИЕ

к выполнению лабораторных работ № 1, 2

по дисциплине

«СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ»

для студентов 3 курса

специальности 220100

дневного обучения

Москва- 2003

Рецензент канд..техн.наук М.М. Константиновский, ген. директор ООО ИТК “Феникс”

Романчева Н.И., канд. техн.наук, доцент

Пособие к выполнению лабораторных работ № 1,2 по дисциплине "Системное программное обеспечение". - М.: МГТУ ГА, 2003.- 40 с.

Данное методическое пособие издается в соответствии с учебным планом для студентов специальности 220100 дневного обучения.

Рассмотрено и одобрено на заседаниях кафедры 15.04.2003г. и Методического совета по специальности 220100 15.04.2003 г.

С О Д Е Р Ж А Н И Е

1 ОСНОВНЫЕ ТРЕБОВАНИЯ И ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Настоящее пособие предназначено для студентов специальности 220100, выполняющих лабораторные работы по дисциплине "Системное программное обеспечение". В пособие включены материалы по лабораторным работам № 1, 2.

Продолжительность каждой лабораторной работы - 4 часа.

Целью проведения лабораторных работ является закрепление основных теоретических положений, изложенных в лекциях на примере широко используемых в различных областях ОС UNIX (ASPLinux).

В процессе выполнения лабораторных работ студенты должны получить практические навыки работы в ОС ОС ASPLinux, в том числе:

- установка и конфигурирование;
- текстовые и графические режимы работы;
- настройка командных оболочек;
- создание учетной записи пользователей и root;
- работа с файловой системой;
- изучение расширенных возможностей командных интерпретаторов;
- получение навыков отладки сценариев.

Лабораторная работа состоит из следующих этапов:

1) домашняя подготовка;

2) выполнение работы на компьютере в соответствии с заданием;

3) сдача выполненной работы преподавателю на персональном компьютере;

4) распечатка результатов работы на принтере;

5) оформление отчета;

5) защита лабораторной работы.

В процессе домашней подготовки студент:

- изучает лекционный материал, материалы по темам данного пособия и дополнительной литературы,
- знакомится с заданием на выполнение лабораторной работы;
- готовит отчет по выполнению лабораторной работы (пункты, отмеченные знаком *).

Выполнение лабораторной работы производится во время занятий в классе ЛВС кафедры ВМКСС МГТУГА в присутствии преподавателя. В процессе выполнения лабораторной работы студент последовательно выполняет задание. По завершению работы - демонстрирует преподавателю результаты.

Сдача работы преподавателю на персональном компьютере заключается в демонстрации выполненной работы и выполнении непосредственно при преподавателе индивидуального задания.

После приема преподавателем лабораторной работы на ПК студент:

- сохраняет результаты лабораторной работы на дискете, выданной преподавателем, в каталоге со своей фамилией;
- распечатывает результаты на принтере на подготовленных листах формата А4.

Отчет по каждой лабораторной работе должен содержать:

- название работы*;
- цель лабораторной работы*;
- задание на выполнение лабораторной работы*;
- краткие комментарии по выполнению лабораторной работы*;
- распечатки файлов результатов, подписанные преподавателем.

Защита лабораторной работы преподавателю проводится по контрольным вопросам и при наличии оформленного отчета (распечатки должны быть приклеены). После защиты лабораторной работы делается соответствующая запись на отчете студента.

2 ЛАБОРАТОРНАЯ РАБОТА №1 ОС ASPLINUX: УСТАНОВКА, НАСТРОЙКА И ИССЛЕДОВАНИЕ СИСТЕМЫ

2.1 Цель работы

Целью данной работы является изучение основных приемов конфигурирования и работы в ОС ASPLinux, а также мониторинга системы.

2.2 Задание на выполнение работы

1) *Выполнить установку и конфигурирование системы. Рекомендованная конфигурация предусматривает:

- наличие загрузчика ASPLinux ASPLoader, установленного в Главную загрузочную запись (MBR),
- автоматическую загрузку графической среды X Windows System и графической оболочки (GNOME),
- существование регистрационной записи обычного пользователя, созданной при установке системы.

2) Запустить корректно установленную систему.
3) Идентифицировать себя с помощью одной из регистрационных записей обычного пользователя.
4) Получить доступ к пользовательским программам.
5) Освоить приемы работы в текстовом и графическом режимах:

- ввод и вывод основных команд;
- перенаправление команд;
- использование каналов;
- файловые операции mc;
- работа со ссылками;
- настройка прав доступа;
- настройка параметров GNOME;

- монтирование сменных устройств средствами GNOME.

6) Выполнить настройку командных оболочек (в соответствии с заданием преподавателя).
7) Провести исследование системы:

- кто находится в системе;
- кто недавно завершил работу.

8) Определить, кто и когда зарегистрировался в системе.
9) Получить информацию о зарегистрированных в настоящий момент пользователях.
10) Определить, находится ли user2 в системе, если нет, то добавить.
11) Оценить стабильность работы и загрузку системы.
12) Просмотреть файл /etc/passwd.
13) Изменить владельца текущего сеанса и просмотреть информацию о процессе.
14) Получить информацию о таблице взаимодействия процессов.
15) Определить количество текстовых и графических консолей. Выполнить переключение между консолями (используя горячие клавиши).
16) Результаты исследований занести в отчет.

ВНИМАНИЕ! Пункт, помеченный *, выполняется группой студентов на одном компьютере под контролем преподавателя.

2.3 Основные приемы работы

Linux - это название ядра операционной системы, базового программного обеспечения низкого уровня, которое управляет аппаратной частью компьютера. В то же время полноценная операционная система включает набор утилит и прикладных программ, который может варьироваться. В зависимости от состава программных компонент выделяют следующие дистрибутивы Linux: Red Hat, Caldera, SuSe, Debian, Slackware.

Рассматриваемая ниже ASPLinux представляет собой версию операционной системы Linux полностью совместимую с Red Hat Linux. ASPLinux - многозадачная, многопользовательская сетевая операционная система. Она поддерживант стандарты открытых систем и протоколы сети Internet. Все компоненты системы, включая исходные тексты, распространяются с лицензией на свободное копирование и установку для неограниченного числа пользователей.

2.3.1 Установка ASPLinux

1) Выбор языка установки

В диалоговом окне Choose installation language (рисунок 1) установите радиокнопку в положение, соответствующее языку, на котором будет осуществляться интерфейс программы установки.

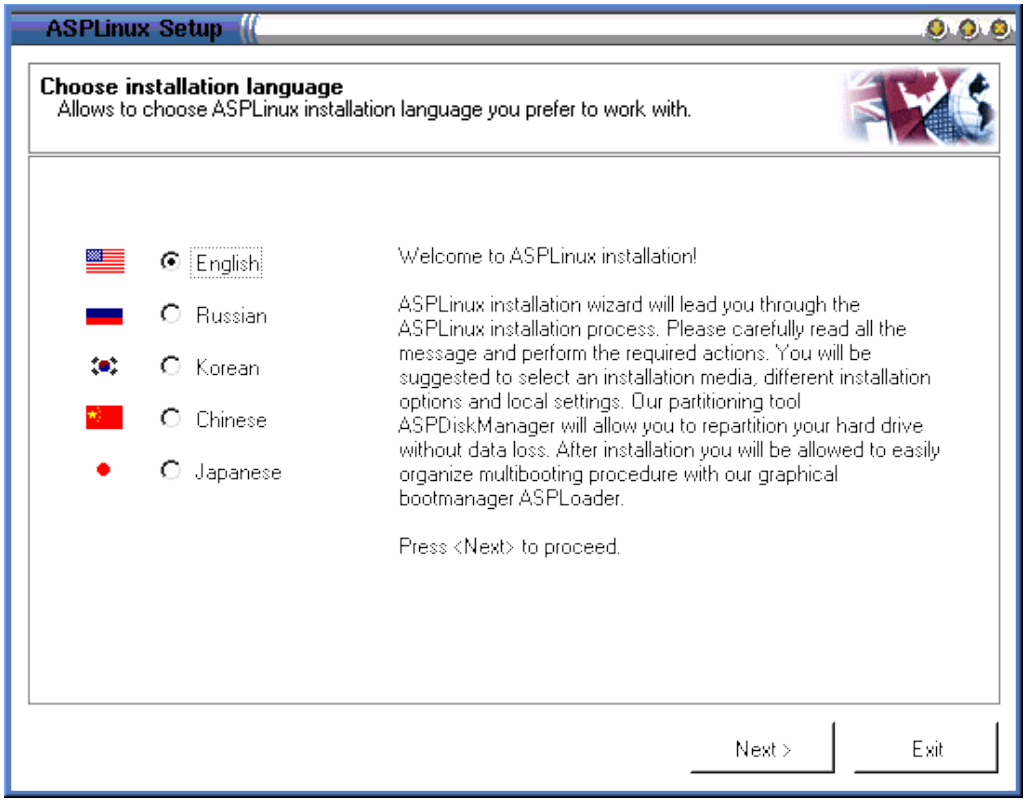


Рисунок 1 - Окно Choose installation language

Выбор языка установки не влияет на то, будет ли установлена локализованная версия ASPLinux и какая именно. Например, можно

воспользоваться русскоязычным интерфейсом программы установки, однако установить нелокализованную (английскую) версию ASPLinux.

При выборе вариант Russian наименование окна меняется на *Выбор языка установки,* все остальные текстовые элементы интерфейса также отображаются на русском языке. Для завершения работы программы установки нажмите на кнопку *Выход*. Для продолжения работы нажмите на кнопку *Далее*.

2) Выбор типа установки

В диалоговом окне Тип установки (рисунок 2) установите радиокнопку в положение, соответствующее одному из типов установки, различающихся подробностью информации, запрашиваемой программой у пользователя:

- быстрая установка (программа самостоятельно принимает решения по большинству вопросов);
- выборочная установка (программа позволяет пользователю самостоятельно определить конфигурацию устанавливаемой системы, а также изменять структуру разделов жесткого диска в ходе установки).

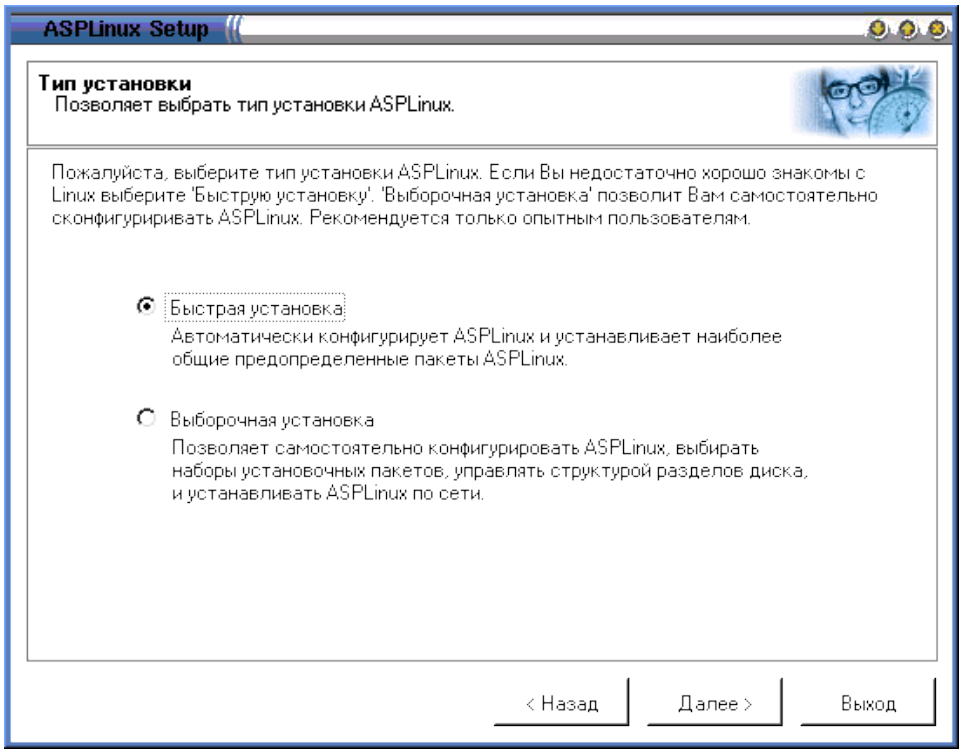


Рисунок 2 - Окно Тип установки

Для завершения работы программы установки нажмите на кнопку *Выход.* Для продолжения работы нажмите на кнопку *Далее*. Для возврата к предыдущему шагу нажмите на кнопку *Назад*

3) Выбор устанавливаемых компонентов

Данный шаг пропускается, если была выбрана *Быстрая установка.*

Для того, чтобы включить все пакеты, входящие в состав одной или нескольких стандартных конфигураций, установите соответствующие флажки:

- *Типовая* - устанавливаются основные компоненты, требуется 450 Мб;
- *Разработчику* - дополнительно устанавливаются средства разработки, требуется 548 Мб;
- *Офис* - требуется 429 Мб;
- *Сервер* - не устанавливается графическая подсистема, требуется 302 Мб.

При этом в информационном окне *Требуется* (MB) показывается дисковое пространство, необходимое для данного состава компонентов, а в информационном окне *Доступно на диске* - дисковое пространство, доступное при выбранном диске и способе использования дисковой памяти.

Для того, чтобы уточнить состав компонентов (пакетов), включаемых в систему, установите флажок *Выборочно*.

В диалоговом окне *Выбор отдельных пакетов* можно просматривать дерево каталогов, содержащих пакеты, используя при необходимости кнопки свертывания и развертывания отдельных ветвей дерева.

Для того, чтобы включить в состав системы отдельный пакет или каталог пакетов, установите флажок, расположенный возле наименования пакета или каталога пакетов. Для того, чтобы исключить пакет или каталог - удалите флажок.

4) Настройка языка интерфейса системы и дополнительных раскладок клавиатуры

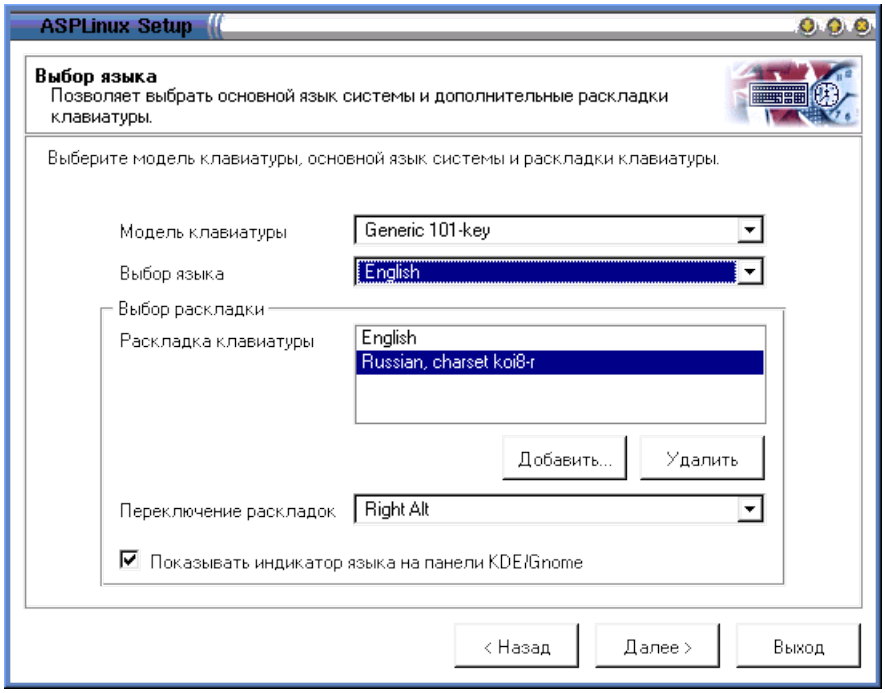
На этом шаге программа установки предлагает выбрать язык интерфейса ASPLinux и установить поддержку раскладок клавиатуры для языков, отличных от американского английского. Следует иметь в виду, что даже при установке локализованной версии системы некоторые компоненты будут использовать англоязычный интерфейс.

В диалоговом окне *Выбор языка* (рисунок 3) выберите из раскрывающегося списка *Модель клавиатуры* клавиатуру, установленную на Вашем компьютере. Из раскрывающегося списка *Выбор языка* выберите основной язык интерфейса.

Для установки дополнительной раскладки клавиатуры нажмите на кнопку *Добавить* и выберите из открывшегося раскрывающегося списка необходимую раскладку.

Для установления клавиши (комбинации клавиш), переключающих раскладки клавиатуры, выберите нужную клавишу (комбинацию) из раскрывающегося списка *Переключение раскладок.*

Рисунок 3- Окно Выбор языка



Для завершения работы программы установки нажмите на кнопку *Выход.* Для продолжения работы нажмите на кнопку *Далее*. Для возврата к предыдущему шагу нажмите на кнопку *Назад*.

5) Настройка даты и времени

На этом шаге можно установить, используя диалоговое окно *Установка даты и времени* (рисунок 4), правильную дату и время и задать временную зону (часовой пояс).

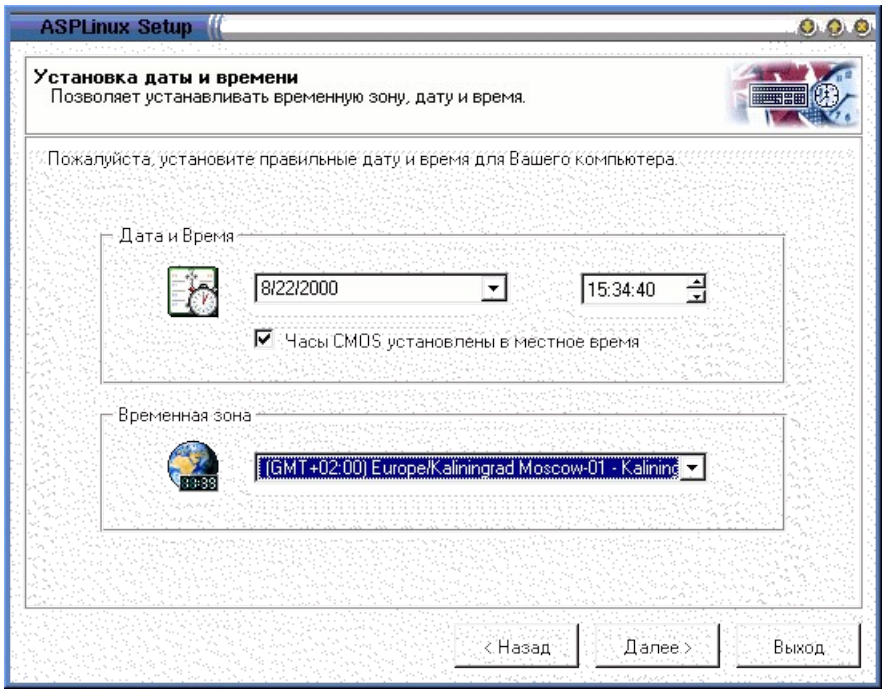


Рисунок 4 - Окно Установка даты и времени

Для завершения работы программы установки нажмите на кнопку *Выход*. Для продолжения работы нажмите на кнопку *Далее*. Для возврата к предыдущему шагу нажмите на кнопку *Назад*.

6) Завершение ввода параметров установки

На этом шаге завершается ввод параметров установки. В информационном окне *Параметры установки* (рисунок 5) приводится краткий список параметров, введенных пользователем на предыдущих шагах.

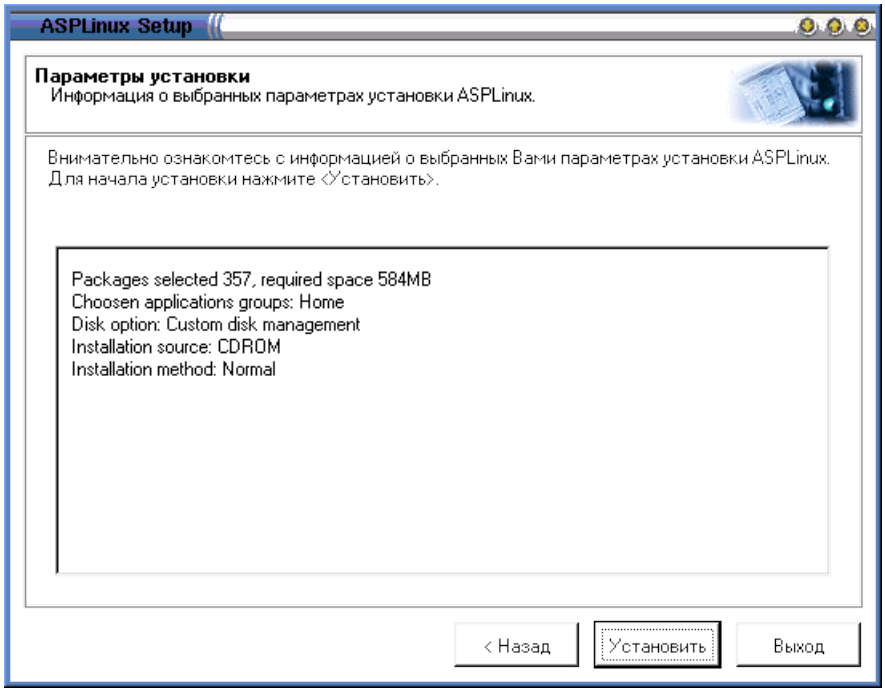


Рисунок 5 - Окно Параметры установки

Для завершения работы программы установки нажмите на кнопку *Выход*. Для продолжения работы нажмите на кнопку *Далее*. Для возврата к предыдущему шагу нажмите на кнопку *Назад*.

7) Создание файловых систем и перенос файлов

На этом шаге программа установки создает основной раздел для размещения файлов ASPLinux и раздел подкачки, форматирует их (создает соответствующие файловые системы) и устанавливает в основной раздел ядро ОС и выбранные компоненты. Никаких действий пользователя на данном шаге не требуется.

8) Настройка монитора

На этом шаге программа установки предлагает определить параметры монитора компьютера, на котором будет работать система. Некоторые мониторы определяются программой установки автоматически, в этом случае пользователю предоставляется возможность согласиться с этим предположением программы или выбрать монитор самостоятельно.

В большинстве случаев выбирается монитор из предлагаемой базы известных моделей.

Для того, чтобы выбрать монитор из базы известных моделей, необходимо выполнить следующие действия:

- в диалоговом окне *Выбор монитора* (рисунок 6) выберите в списке *Изготовитель* наименование фирмы-производителя Вашего монитора;
- в появившемся списке *Модель* (наименованиями известных моделей данного производителя) выберите нужную модель.

В случае отсутствия в списках нужной модели или нужного производителя, выберите в списке *Изготовитель* наименование *Custom* и установите вручную частоты вертикальной и горизонтальной развертки.

Для того, чтобы установить значения частот вертикальной и горизонтальной развертки, выберите из раскрывающихся списков *Горизонтальная развертка* и *Вертикальная развертка* значения, соответствующие данному монитору. Если эти параметры монитора неизвестны, выберите наименьшие значения из имеющихся в списке.

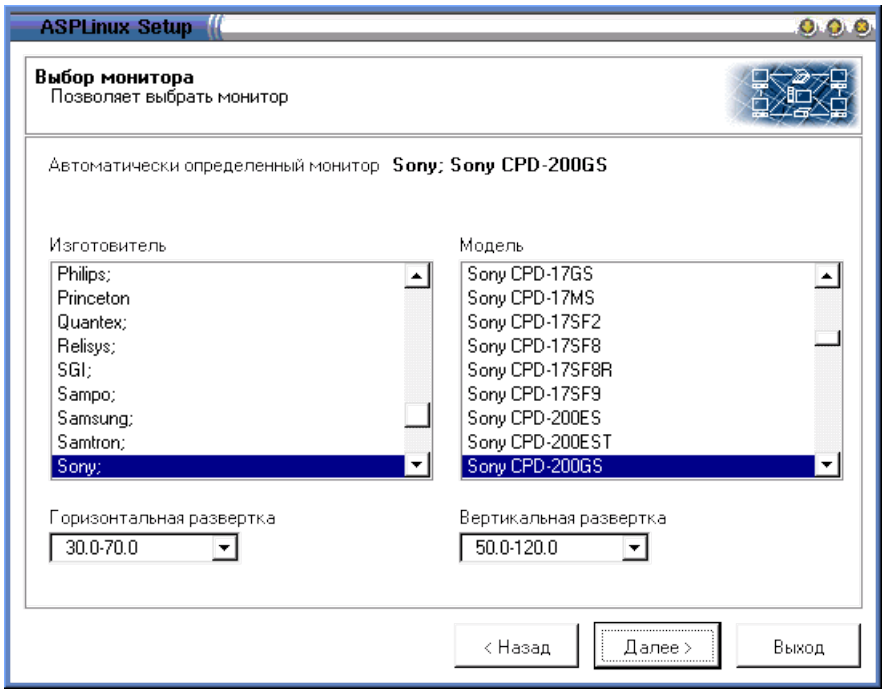


Рисунок 6 - Окно Выбор монитора

Для завершения работы программы установки нажмите на кнопку *Выход*. Для продолжения работы нажмите на кнопку *Далее*. Для возврата к предыдущему шагу нажмите на кнопку *Назад*.

9) Завершение работы программы установки

Программа установки завершает свою работу и выдает на экран окно *Установка ASPLinux* успешно завершена. Нажмите на кнопку *Перезагрузка.* Компьютер перезагрузится.

Если установка производилась в режиме загрузки компьютера с дистрибутивного компакт-диска, не забудьте удалите компакт-диск из привода и восстановить порядок загрузки компьютера с жесткого диска.

Для того, чтобы установить режим загрузки компьютера с жесткого диска:

- включите или перезагрузите компьютер;
- запустите программу настройки параметров BIOS, следуя рекомендациям документации к данному компьютеру или его материнской плате (в большинстве компьютеров для этого необходимо нажать определенную клавишу, указанную в экранной подсказке, сразу после включения компьютера или после теста оперативной памяти);
- установите в параметрах BIOS первоочередную загрузку с жесткого диска, сохраните измененные параметры BIOS.

10) Задание пароля root

Ввод пароля root (суперпользователя, администратора системы) осуществляется два раза (для проверки правильности ввода), из соображений секретности вводимый пароль отображаться не будет. Пароль должен быть достаточно длинным, минимальная длина - 6 знаков, содержать как прописные, так и строчные символы, буквы и цифры. Приветствуется наличие в пароле таких символов как пробел, дефис, подчеркивание, знаки препинания.

2.3.2 Запуск компьютера

После включения или перезагрузки компьютера запускается программа ASPLoader . Эта программа по выбору пользователя запускает любую из ОС, инсталлированную на данном компьютере.

2.3.3 Идентификация пользователя

Для идентификации пользователя:

- введите в поле ввода *Login:* регистрационное имя обычного пользователя.

После ввода имени пользователя система выполняет аутентификацию, для чего требуется ввести пароль, соответствующий данному пользователю.

- введите в поле *Password:* пароль пользователя.

Если пароль или имя пользователя введены неправильно, система потребует снова ввести имя пользователя и пароль. При корректном имени и пароле запускается оболочка GNOME.

Если при установке системы не было создано ни одной регистрационной записи обычного пользователя, введите имя суперпользователя *root* и добавьте соответствующую запись.

2.3.4 Добавление регистрационной записи обычного пользователя

Для того, чтобы добавить регистрационную запись обычного пользователя, необходимо:

- перейти в текстовую консоль или режим эмуляции терминала и войти в систему как суперпользователь;

- в поле ввода *Имя пользователя* ввести имя пользователя, состоящее из маленьких латинских букв и цифр и начинающееся с буквы;

- в поле ввода *Полное имя* ввести обычное имя пользователя, записанное латинскими буквами (например, Ivan Petrovich Smirnov);

- в поле ввода *Пароль* ввести пароль этого пользователя;

- в поле ввода *Подтвердите пароль* ввести этот же пароль повторно, далее нажать на кнопку *Добавить*.

Команда добавления пользователя *adduser* создает регистрационную запись для нового пользователя (для чего вносит изменения в ряд конфигурационных файлов системы) и создает так называемый домашний каталог */home/имя_пользователя*, содержащий некоторые необходимые файлы и подкаталоги. Этот каталог и находящиеся в нем файлы принадлежат пользователю, и он имеет полный набор прав для работы как с существующими, так и вновь создаваемыми объектами в этом каталоге.

Рекомендуется создать хотя бы одного обычного пользователя. Не рекомендуется пользоваться правами суперпользователя без необходимости.

2.3.5 Удаление регистрационной записи обычного пользователя

Чтобы удалить ошибочно созданную регистрационную запись:

- необходимо работать с правами суперпользователя;
- ввести в командной строке команду *userdel имя _пользователя*;

Для удаления регистрационной записи, а также принадлежащих пользователю файлов, необходимо ввести в командной строке команду

userdel - r имя _пользователя

Эта команда удаляет каталог */home/имя_пользователя* и все его содержимое.

Удаление регистрационной записи пользователя невозможно, если он в данный момент зарегистрирован в системе или работает какой-либо процесс, запущенный от его имени.

2.3.6 Текстовые и графические режимы работы ASPLinux

В ходе загрузки системы создаются несколько консолей - виртуальных устройств ввода-вывода, которыми могут пользоваться различные компоненты и пользовательские программы системы. ASPLinux в стандартной настройке работает с 7 виртуальными консолями (6 текстовых и 1 графическая), их которых в каждый момент времени

только одна может быть связана с реальной (физической) консолью, т.е. является активной. Консоли, представляющие информацию только в текстовом виде с использованием экранных шрифтов в форматах видеосистемы компьютера, называются *текстовыми*. В таких консолях используется интерфейс командной строки. Другие консоли (*графические*) представляют информацию в графическом виде, используя Графический пользовательский интерфейс (GUI). Как правило, в одной из консолей автоматически запускается графическая среда X Windows System и графическая оболочка GNOME. Для перехода между консолями используется сочетание клавиш [CTRL]+ [ALT]+ [Fn], n - номер консоли, находится в интервале от 1 до 12. Например, чтобы сделать активной консоль с номером 4, следует нажать клавиши [CTRL]+ [ALT]+ [F4].

2.3.7 Обзор команд

Синтаксис многих команд предусматривает использование параметров довольно сложного формата, указываемых в командной строке после имени команды. Полное описание команд и их параметров можно получить, набрав *man имя_команды*. Дополнительную информацию по команде - *info имя_команды*.

Ниже приведены наиболее употребительные команды и наиболее частые форматы.

а) команды управления файлами

ls [opt] [file1 file2 …] - вывод имен файлов текущего каталога. В качестве параметров можно задать имена каталогов, содержимое которых нужно вывести, или имена файлов, информацию о которых нужно получить. Опции команды позволяют получить список дополнительной информации:

ls - список файлов текущего каталога (краткий формат).

ls -al - получить список файлов текущего каталога с указанием размера, времени создания и изменения, имени владельца, таблицы прав и других данных, например:

-rwxr- - r - - 2 nata group 34 Nov 10 10:34 a.out

где *-rwxr--r--* - права доступа (за исключением первого символа, обозначающего тип файла: - – обычный файл, d – каталог, p – именованный канал, b – специальное блочное устройство, c – специальное символьное устройство) на чтение (read -символ r), запись (write -символ w), выполнение (execute - символ x). Наличие прав обозначается соответствующим символом, а отсутствие - символом "-";

2 – число жестких связей (hard link) данного файла;

nata - имя владельца -пользователя (user owner) файла. Владельцем-пользователем вновь созданного файла является пользователь, запустивший процесс, который и создал файл;

group – имя владельца - группы (group owner). Порядок назначения владельца группы зависит от конкретной версии UNIX;

34 - размер файла;

Nov 10 - дата последнего изменения;

10:34 - время последнего изменения;

a.out - имя файла.

ls –aC – просмотр скрытых файлов**;**

ls -C *nata* **-** вывод списка файлов каталога *nata* в несколько колонок в алфавитном порядке;

ls –RC */home/nata/bin* **-** рекурсивный просмотр каталогов, например, */home/nata/bin*;

ls –tC - сортировка по времени модификации, все вновь созданные файлы размещаются в начале списка;

ls –ctC - сортировка по изменению статуса (изменение владельца или прав доступа). Если ключ **t** не задан, то ключ **c** игнорируется.

cd [dir] - сменить текущий каталог. При задании без параметра – происходит переход в домашний каталог пользователя;

cd файл1 файл2 - копировать файл. Если вместо имени второго файла указать каталог, то **файл1** копируется в каталог **файл2** с тем же именем,

при этом в имени первого файла допускается использование подстановочного символа “звездочка”;

rm файл1 – удалить файлы с указанными именами. Допускается использование подстановочного символа “звездочка” и другие специальные возможности. Например, команда *rm *m** позволит удалить все файлы, в именах которых встречается буква *m;*

mkdir [имя_каталога1]… – создать новый каталог;

rmdir [имя_каталога]… – удалить пустой каталог;

ln [-опция] source target - создает жесткую связь имени *source* с файлом, адресуемым именем *target*. При использовании опции *–s* будет создана символическая ссылка;

pwd - вывести имя текущего каталога;

cmp [-опция] файл1 файл2 - сравнить два файла, указанных в качестве аргумента. Если файлы одинаковы, то никакое сообщение не выводится, в противном случае выводятся данные о первом несоответствии между этими файлами, например:

file1 file2 differ: char 15, line 6 ,

найдено различие в 15 символе 6-ой строки.

б) управление выводом на экран

cat [-опция] файл - выводит содержимое *файла* на экран терминала. Использование ключа *–v* целесообразно при просмотре нетекстового файла. В этом случае вывод “непечатных ” символов, которые могут нарушить настройки терминала, будет подавлен;

more [-опция] файл – выводит стандартный входной поток на экран порциями по 24 строки, ожидая нажатия клавиши *Пробел* для вывода очередной порции. Досрочно завершить вывод можно, нажав клавишу Q;

less - выводит стандартный входной поток на экран порциями по 24 строки, ожидая нажатия клавиши *Пробел* для вывода очередной порции. В отличии от команды *more* поддерживает возможность прокрутки вверх и поиска;

head [-n] файл – просмотреть только начало (первые n строк) файла;

tail [-опция] файл – просмотреть конец (последние n строк) файла;

в) поиск файлов

find имя_каталога [-ключ] - выполнить поиск файла в файловой системе, начиная с каталога ***имя_каталога***, используя различные критерии:

- name – поиск по искомому имени файла, например:

find / -name sh ,

по этой команде будет осуществляться поиск в каталоге / файла с именем *sh;*

-print – обеспечивает вывод информации. Например, для вывода полного имени исполняемого файла командного интерпретатора Bourne shell, необходимо ввести команду:

find / -name sh -print 2 >/dev/null ;

Для фрагментарного поиска по имени файла (только в последней части спецификации файла), например, '*core*', следует ввести команду:

find ~ -name '*core*' *-print*

- size [размер] – поиск по заданному размеру. Например, для поиска файлов размером больше 10 Мбайт по всей файловой системе, необходимо ввести команду:

find . –size +20480 - print ;

- atime - поиск по последнему времени модификации. Например, поиск файлов с именем *file1,* обращение к которым было более *15* дней назад:

find / –name file1 -atime +15 - print ;

Для автоматического удаления всех найденных файлов с именем *core* (образ процесса, создаваемый при неудачном его завершении и используемый в целях отладки), последнее обращение к которым было более месяца (*+30*) назад, следует ввести команду:

find / –name core -atime +30 - exec rm {} \ ;

Следует отметить, что каждый раз при запуске команды, указанной после ключа *exec*, создается новый процесс. Это приводит к увеличению нагрузки на систему и излишнему потреблению ресурсов процессора и оперативной памяти. Однако при необходимости выполнить операцию, например такую как *rm,* над большим количеством файлов, эффективнее сначала построить список файлов, а затем запустить команду *rm* лишь один раз, передав ей этот список в качестве параметра.

Команда *find . –print* аналогична команде *ls –Rfl,* но в последнем случае выводимый список будет длиннее, т.к. в процессе обхода команда ls отмечает каждый новый каталог, а команда find не обращает внимание на каталог **..** ;

which [-ключ] - поиск выполняемых файлов. Данная команда встроена в оболочку, позволяет определить точное местонахождение файла, и передает результаты своего выполнения в стандартный выходной поток. В оболочке C команда *which* позволяет определить, какие из команд являются встроенными, а какие псевдонимами.

г) генерация отчетов и обработка результатов других команд

grep [-ключ] <рег_выражение> файл1… – команда со сложной структурой параметров. Позволяет выбрать из текстового файла (обычно созданного в результате работы других команд через канал) строки по сколь угодно сложным критериям. Как правило, эти строки передаются через канал для дальнейшей обработки. Например, чтобы произвести поиск в файле *file* слова “центр” в американском (*center*) и британском (*centre*) написании, можно задать следующую команду:

grep “cent[er]” file

или

grep “cent[er] [er]” file ,

где *[er]* является регулярным выражением, соответствующим либо символу “e”, либо ”r”. Регулярное выражение должно быть заключено в

кавычки для предотвращения интерпретации специальных символов командным интерпретатором shell.

Если подстрока уже содержит кавычки, их надо экранировать, поместив символ "\" непосредственно перед кавычками. Например:

grep "лекция по дисциплине \"СПО\"" file

Для выполнения поиска, нечувствительного к заглавным/строчным символам, необходимо использовать ключ *–y*. Для поиска строк, не содержащих указанную подстроку, используется ключ *–v*.

Пример команды с более сложной структурой:

ps -ef |grep mproc - получить информацию о конкретном процессе *mproc* и отфильтровать поток, оставляя лишь строки, в которых есть слово *mproc;*

д) мониторинг системы

Для управления дисковым пространством в UNIX используются команды *df, du* и *ulimit*:

df [-ключ] – команда определяет, сколько свободного дискового пространства и индексных дескрипторов доступно в разделе смонтированного диска.

По умолчанию команда используется без параметров и выводит объем свободного пространства, например:

/ (/dev/hdb1): 260836 blocks 12034 files

/home (/dev/sda1): 260836 blocks 2104 files

В первом столбце содержится точка монтирования данной файловой системы. Затем в круглых скобках следует имя смонтированного физического устройства (в UNIX все устройства являются файлами, даже сама файловая система). Следующий столбец отображает число свободных блоков размером по 512 байт. В последнем столбце выводится количество файлов, содержащихся на данном устройстве.

При использовании ключей:

-k – вывод данных осуществляется в блоках по 1024 байт, или в килобайтах. При этом данные выводятся в формате, принятом в системе BSD:

Filesystem	1024- blocks	Used	Available	Capacity	Mounted on
/dev/hdb1	*1112646*	*972611*	*140035*	*88%*	*/*
/dev/sda1	*961374*	*720104*	*241270*	*75%*	*/home*

В первом столбце указано имя устройства, на котором расположена файловая система. Во втором столбце отображается размер файловой системы в блоках по 1 Кбайт. В третьем столбце выводится число используемых блоков, а в четвертом – число свободных блоков. В пятом столбце выводится процент использования диска. В последнем столбце указывается точка монтирования системы;

-P – информация отображается в формате, определенном в стандарте POSIX, который аналогичен формату, принятому в BSD;

-t - информация отображается в формате, который близок к стилю, используемому в SYSTEM V. Данные выводятся в блоках размером по 512 байт, кроме того, приводится информация, как о количестве блоков, так и о количестве индексных дескрипторов;

-i - предназначен для подсчета количества индексных дескрипторов (не поддерживается стандартом POSIX). Выводимая информация имеет следующий вид:

Filesystem	Inodes	IUsed	IFree	%IUsed	Mounted on
/dev/hdb1	*301056*	*93059*	*207997*	*31%*	*/*
/dev/sda1	*260096*	*17280*	*242816*	*7%*	*/home*

В качестве параметров команде *df* можно передать имя файла или список имен файлов. В этом случае отображается информация только о тех файловых системах, которые содержат указанные файлы.

du [- ключ] - команда определяет какой объем диска занимает конкретный каталог. Вызов команды без параметров позволяет получить данные о текущем каталоге. Если в качестве параметра указать имя

каталога, то будет отображена информация обо всех каталогах, расположенных в иерархии ниже текущего. Если в качестве параметра указано имя файла, не являющееся каталогом, то не выводится никакой информации.

Команда *du* имеет четыре ключа:

-k – имеет то же значение, что и для команды *df* , при этом данные об использовании дискового пространства представляются в килобайтах;

-a – задает вывод данных всех перечисленных файлов. При этом полученный результат аналогичен результатам выполнения команды *ls – ls*;

-s – задает ограниченный вывод, только данные об указанном каталоге, например: *13500 /home/nata/bin,* где 13500 – размер каталога, выраженный в блоках по 512 байт;

-x – не выводятся данные о файлах, находящихся в других файловых системах. Таким образом проверяются данные, хранящиеся в указанном каталоге локального диска;

ulimit – выводит или устанавливает значение пределов, ограничивающих использование задачей системных ресурсов (времени процессора, памяти, дискового пространства);

top – команда выдает непрерывно обновляемую таблицу всех задач, выполняющихся на компьютере, включая системные, с указанием объема используемых ресурсов. Для завершения работы команды необходимо нажать клавишу *Q*;

ps – выводит информацию о существующих процессах**.** При использовании различных опций можно получить следующую информацию:

-al - выдает в форме таблицы список пользовательских процессов, запущенных в системе;

-F – статус процесса (системный, блокировки памяти и т.д.);

-A – состояние всех процессов;

-S – состояние процесса (O – выполняется процессором, S – находится в состоянии сна, R - готов к выполнению, I - создается, Z - зомби);

- ef – распечатывает имя программы, породившей процесс, вместе со всеми параметрами;

- n name – состояние всех процессов, порожденных командами, имена которых указаны в списке *name*;

- g list – показать все процессы, запущенные пользователями групп, номера которых указаны в списке. Например, *ps –g 0* -.показать все процессы группы 0, т.е. root. Номера групп указываются в списке через запятую или пробел.

- l – длинный формат вывода состояния процессов;

- p - состояние процессов, идентификаторы которых указаны в списке, например: *ps –p "12499, 17772"* – определить состояние процессов с идентификаторами (PID) 12499 и 17772;

w [- ключ] – команда информирует о том, что делают в системе зарегистрированные пользователи, например:

(9:12 am up 30 min, 3 users, Load average, 0.00, 0.52, 1.22)

user	*TTY*	*FROM*	*LOGIN@*	*IDLE*	*JCPU*	*PCPU*	*what*
user	*tty1*	**-**	*8.44 am*	*27:50*	*0.24s*	*0.03s*	*/bin/sh/usr*
user	*pts/0*	**-**	*8.52 am*	*29:48*	*0.00s*	*?*	*-*

Первая строка содержит текущее время, сколько времени компьютер работает без перезагрузки, число пользователей и загрузка машины. Затем следует строка, содержащая заголовки столбцов: *user* – имя пользователя, связанного с данным устройством tty; *TTY* - имя терминала (консоли); *LOGIN@* -первоначальное время регистрации; *IDLE* - количество времени, на протяжении которого пользователь ничего не вводил с клавиатуры; *ICPU* – общее время центрального процессора, использованного всеми процессами на этом терминала; *PCPU* – общее время центрального

процессора для всех активных процессов на этом терминале; *what* – название и параметры текущей выполняемой команды. Далее следует список пользователей, и чем они заняты. Знак ? означает, что процесс ожидает связи с терминалом, однако в текущий момент связь отсутствует. Команда имеет три ключа:

- h – подавляет заголовки;

- l – отображает информацию в расширенном виде (используется по умолчанию);

- s – отображает информацию в краткой форме (выводятся столбцы *user, tty, iIdle, what*);

Конкретного пользователя можно проверить, введя команду

w имя _пользователя

who [-ключ]– выдается список пользователей, зарегистрированных в данный момент в системе. Например:

nata tty1 Nov 2 14:30

alex tty4 Nov 2 14:15

где - *nata* – имя пользователя, *tty1* - номера его терминала, *Nov 2* - дата и *14:30* - время подключения.

Согласно стандарту POSIX, команда должна иметь несколько ключей, влияющих на внешний вид выводимой информации:

-b – выводит время последней перезагрузки;

-d – выводит список “умерших” процессов (dead processes), которые не были повторно порождены;

-H – выводит заголовки столбцов;

-l – перечисляет номера tty, ожидающих регистрации пользователей;

-T – выводит состояние канала связи с каждым из терминалов (+ означает, что данный терминал доступен для записи, а – означает, что терминал для записи не доступен);

-t – выводит момент последнего изменения системного времени;

-s – выводит имя пользователя, tty и время регистрации в системе (используется по умолчанию);

-u – выводит время простоя для каждого терминала;

-m – выводит информацию только о текущем терминале;

-r – выводит текущее состояние системы;

-p – перечисляет все активные процессы, порожденные процессом *init;*

-g – перечисляет только пользовательские имена и количество пользователей;

Пример результата выполнения команды *who –THu :*

USER	*MESG*	*LINE*	*LOGIN-TIME*	*IDLE*
nata	+	*tty1*	*nov 10 18:44*	.
oleg	-	*tty3*	*nov 10 19:53*	*old*
alex	+	*tty4*	*nov 10 18:53*	*old*

Из примера видно, что только пользователь *nata* находится в активном состоянии. Пользователи *oleg* и *alex* не обращались к своим терминалам на протяжении дня. Кроме того, пользователю *oleg* доступ к терминалу запрещен;

last [-ключ] – позволяет определить, кто и когда зарегистрировался в системе. Для выдачи результатов она пользуется файлом /etc/utmp, в котором зафиксированы моменты входа-выхода пользователей и перезагрузки системы. При использовании команды без параметров будет выведен список в обратном порядке всех, кто работал в системе. Для ограничения размера списка в качестве параметра следует указать некоторое число, например, команда *last* -25 выводит список последних 25 пользователей. Введя команду *last reboot,* можно просмотреть список последних перезагрузок;

finger – команда позволяет определить, находится ли в системе некоторый пользователь. Введя команду *finger – имя_пользователя,* можно получить

разнообразную информацию, включающую и время последней регистрации данного пользователя в системе;

at [-ключ] время_запуска - считывает команды стандартного потока ввода и группирует их в задания *at,* которое будет выполнено в указанное пользователем время. . Например:

at now + 2minutes

Для выполнения задания будет запущен командный интерпретатор, в среде которого и будут исполнены команды.

uptime – позволяет оценить стабильность и загрузку системы. Данная команда выводит только первую строку информации команды *w*, например,

9:12 pm up 10 days, 10:51, 4 users, load average: 0.01, 0.03, 0.22)

kill [-sig] pid1, pid2, .. – посылает процессам с идентификаторами *pid1, pid2*,…сигнал *sig*, что вызывает принудительное прекращение процесса. Параметр *pid* может быть либо идентификатором процесса, либо идентификатором задания (в этом случае перед идентификатором должен стоять символ %). Номер процесса следует посмотреть в столбце *PID,* таблицы, выдаваемой командой *ps – al.* Сигнал *sig* может быть указан как в числовой, так и символьной форме. Команда **kill –l** выводит таблицу соответствия между символьными именами сигналов и их числовыми значениями:

1) SIGHUP	2)SIGINT	3)SIGQUIT	4)SIGILL
5) SIGTRAP	6)SIGIOT	7)SIGBUS	8)SIGFPE
9) SIGKILL	10)SIGUSR1	11)SIGSEGV	12)SIGUSR2
13) SIGPIPE	14)SIGALRM	15)SIGTERM	16)SIGCHLD

…

Таким образом, следующие две команды эквиваленты:

kill –9 13456

kill –SIGKILL 13456

2.4 Вопросы к защите лабораторной работы

1) Перечислите этапы настройки ASPLinux.
2) Особенности работы в текстовой и графической консолях.
3) Структура файловой системы ASPLinux.
4) Как осуществляется монтирование устройств? Приведите формат команды монтирования/размонтирования устройств.
5) Перечислите пользователей системы. Как добавить пользователя, группу пользователей в систему?
6) Какая команда используется для изменения владельца текущего сеанса?
7) Расшифруйте запись из файла /etc/passwd :
   nata:0/RFTGYHcyFf:69:10:Nata_F._Ferri: /home/nata:/bin/csh
8) Перечислите файлы, относящиеся к служебным учетным записям UNIX.
9) Укажите формат команд, используемых для исследования системы.
10) Перечислите команды для идентификации файлов.
11) Укажите формат команды для получения информации о процессах, связанных с терминалом.
12) Расшифруйте поля файла /etc/group: user::10: dave, lory, james .
13) Напишите команду для просмотра таблиц процесса.
14) Перечислите способы установки командного интерпретатора.
15) Назначение файла /etc/shell.
16) Где и в каком поле записи указывается командный интерпретатор пользователя?

2.5 Литература

1) А. Робачевский Операционная система UNIX.-СПб.:BHV-Санкт-Петербург, 1997/2002- 528 с.

2) Команды Linux. Справочник.- К.: ТИД ДС, 2002 - 688 с.

3 ЛАБОРАТОРНАЯ РАБОТА №2 РАСШИРЕННЫЕ ВОЗМОЖНОСТИ КОМАНДНЫХ ИНТЕРПРЕТАТОРОВ

3.1 Цель работы

Целью данной работы является изучение расширенных возможностей командных интерпретаторов, получение навыков отладки сценариев.

3.2 Задание на выполнение работы

1) Используя утилиту at передать в назначенное время сообщение произвольного содержания.
2) Вывести информацию об остановленных и фоновых заданиях:
   - полный список,
   - с указанными номерами заданий,
   - идентификатор группы процессов и рабочий каталог;
   - только идентификатор группы процессов.
3) Выполнить поиск файлов по следующим критериям:
   - по размеру,
   - с именем core (образ процесса, создаваемый при неудачном его завершении и используемый в целях отладки).
4) Изучить интерфейс и выполнить основные файловые операции в встроенном редакторе программ Midnight Commander (mc).
5) Написать и отладить программу на языке интерпретатора Borne:
   - используя команду echo;
   - используя команду управления форматированием вывода (printf);
   - обрабатывающую ввод из потока STDIN;
   - использующую несколько переменных в аргументе команды read, рассмотреть случаи:
     - число переменных в списке равно числу аргументов, считанных из потока;

- число переменных в списке меньше числа аргументов, считанных из потока;
- вычисления длины окружности и площади круга (используя команду bc);

6) Написать и отладить программу, демонстрирующую использование функции setuid (процесс может изменять значение кода идентификации пользователя, под которым он исполняется).
7) Запустить программу ведения журнала logger, вывод программы записать в файл file.log, сохраняя все предыдущие записи.

3.3 Краткие теоретические сведения

3.3.1 Менеджер файлов Midnight Commander (mc)

Программа Midnight Commander полифункциональный менеджер файлов, работающий в текстовом режиме (т.е. в текстовой консоли или терминале). Интерфейс программы похож на двух панельные менеджеры файлов Norton Commander для MS-DOS, FAR и Windows Commander для Windows, а по набору функций не уступает лучшим из них. Файловые операции mc выполняются аналогично.

3.3.2 Работа со ссылками

Символическая ссылка создается в каталоге, открытом на пассивной панели, и указывает на текущий файл на активной панели.

Для того чтобы создать символическую ссылку, необходимо выполнить следующие действия.

1) Перейдите в каталог, в котором будет размещаться ссылка.
2) Нажмите на клавишу *Tab*, чтобы сделать эту панель пассивной.
3) На активной панели перейдите в каталог, где находится файл, на который будет указывать ссылка, и выберите его (текстовым курсором или щелкните мышью).
4) Выберите меню *Файл* (клавиша F9).
5) В этом меню выберите пункт *Символическая ссылка*.

6) В открывшемся окне *Символическая ссылка* введите в поле ввода *Имя символической ссылки* имя файла ссылки и нажмите на клавишу *Enter*.

3.3.3 Простейшие текстовые редакторы

Для работы в текстовой консоли ASPLinux можно воспользоваться несколькими простейшими текстовыми редакторами, которые позволяют изменить конфигурационный файл системы или набрать текст сценария. В текстовом режиме, как и в оболочке GNOME, можно использовать профессиональную систему подготовки текста *emacs* (включающую в качестве макроязыка язык программирования высокого уровня), однако ее рассмотрение выходит за пределы данного пособия.

В простых случаях можно воспользоваться встроенным редактором программы *Midnight Commander (mc)*. Для того чтобы отредактировать текстовый файл во встроенном редакторе *mc*, выберите нужный файл в активной панели и нажмите клавишу *F4*. В открывшемся окне редактора можно вводить или редактировать текст. При необходимости следует использовать кнопки операций с блоками текста или поиска по образцу или, нажав клавишу *F9,* открыть меню, позволяющее устанавливать пользовательские настройки редактора, или осуществлять такие операции, как форматирование текста и обработка при помощи макросов.

Существующий несколько десятков лет тестовый редактор *vi* имеет очень специфическую систему команд и сохраняется в современных системах UNIX (Linux) во многом лишь по традиции. Однако, некоторые старые командные файла (скрипты), могут по умолчанию вызывать данный редактор для редактирования файлов пользователя. В этом случае понадобиться выйти из текстового редактора *vi* без сохранения изменений: поместить курсор с помощью клавиши *Backspace* в ту часть окна, где расположен текст; далее набрать символ *:* (нажав клавиши *Shift* - *:*), курсор вместе с набранным символом переместиться в нижнюю строку

экрана (поле команд); ввести в этом поле последовательность символов *q!* и нажать клавишу *Enter*.

3.3.4 Программирование на языке командного интерпретатора

Интерпретатор (shell) UNIX представляет собой интерфейс командной строки. По своей функциональности он похож на интерпретатор COMMAND.COM системы MS DOS. Одной из задач интерпретатора является обеспечение безопасного и структурированного доступа к ядру UNIX, т.е. фактически это программный уровень, который предоставляет среду для ввода команд, обеспечивая тем самым взаимодействие между пользователем и ядром операционной системы. Кроме того, встроенный в интерпретатор мощный язык программирования используется для решения различных задач: от автоматизации повторяющихся команд до написания сложных интерактивных программ обработки данных, получения информации из небольших баз данных.

В среде UNIX существует много командных интерпретаторов, среди которых можно выделить такие как sh, tcsh, ksh, csh, bash, в Linux по умолчанию используется bash.

Ниже приводятся примеры нескольких способов написания простой программы вывода фразы "Hello, ASRLinux!" на языке интерпретатора Bourne.

Пример 1.

```
#!/bin/sh
# Первая программа "Hello, ASRLinux!",
# реализованная для интерпретатора Bourne
echo
echo   " Hello, ASRLinux! "
echo
exit   0
```

Пример 2.

```
#!/bin/sh
# Программа, "с командой printf",
# реализованная для интерпретатора Bourne
printf    "\n" с командой printf\n\n!"
exit   0
```

Пример 3.

```
#!/bin/sh
# Программа, "воспринимающая ввод с клавиатуры",
# реализованная для интерпретатора Bourne
echo
echo   -n   "Введите name:    "
read   name
echo
echo   "Hello,    ${name}!"
echo
exit   0
```

В примерах 1-3 первые строки содержат последовательность символов #!, сообщающую ОС, что за ней следует имя командного интерпретатора, в котором сценарий должен выполняться. В данном случае это интерпретатор Bourne, /bin/sh. Строки, начинающиеся с символа #, являются комментариями, и интерпретатор их игнорирует. Команда *echo* выводит свои аргументы в стандартный вывод (обычно это экран). Команда *exit* завершает работу программы и возвращает код выхода родительской программе (обычно это командный интерпретатор). Код завершения, равный 0, указывает, что программа нормально завершила работу. Код, отличный от 0, сообщает об ошибке. Если статус завершения не указан явно, то программа возвращает код завершения последней выполненной команды.

Простого вывода сообщений на экран недостаточно для решения

задач, поэтому язык сценариев использует переменные. Все переменные в программах командного интерпретатора хранятся как строки. Кроме присвоения значений переменным и их использования в сценарии, командный интерпретатор предоставляется возможность обрабатывать ввод из потока STDIN. Для чтения пользовательского ввода используется команда *read.* Команда *read* в качестве аргументов может иметь несколько переменных (пример 4). В этом случае всякий пустой символ трактуется как символ разделитель между значениями, присваиваемыми разным переменным. Если число аргументов, прочитанных из потока ввода, меньше, чем число переменных в списке, оставшимся переменным не присваивается никаких значений. Если аргументов больше, чем переменных, то все оставшиеся значения присваиваются последней переменной.

Пример 4.

```
#!/bin/sh
echo
echo -n "Введите три числа, разделенные пробелами или символом табуляции: "
read  var 1  var2  var3
echo
echo  "The of var1 is:  ${var1}"
echo  "The of var2 is:  ${var2}"
echo  "The of var3 is:  ${var3}"
echo
exit  0
```

Для операций с числами с плавающей запятой, а также обработки сложных выражений, где порядок операций изменен, применяется команда *bc* (пример 5).

Пример 5.

```
#!/bin/sh
```

```
# Данная программа вычисляет длину окружности
# и площади круга
pi="3.14159265"         # Переменной присваивается число p
# пользователю выводится сообщение и запрашивается радиус
# окружности
echo
echo  "This program computes both the circumference and the area of"
echo  "a circle"
echo
echo  -n   "please enter radius of tfe circle:  "
read   radius
# Вычисляемые значения сохраняются в переменных
circumference=$(echo   $radius*$pi   | bc  -l)
area= $(echo  $radius^2*$pi   | bc   -l)
#Программа выводит результаты и завершает работу
printf   "\n\nThe circumference is:\t$circumference\n"
printf   "The area is:\t\t$area\n\n"
exit  0
```

После ввода текста программы и сохранения, необходимо сделать файл выполняемым. Для этого используется команда: *chmod u+x hello.*

Эта команда устанавливает права владельца на запуск файла, например, *hello.* Для запуска файла наберите в командной строке: *. /hello* .

3.3.5 Программа setuid

Ядро связывает с процессом два кода идентификации пользователя, не зависящих от кода идентификации процесса: реальный (действительный) код идентификации пользователя и исполнительный код или *setuid* (от "set user ID" - установить код идентификации пользователя, под которым процесс будет исполняться).

Реальный код идентифицирует пользователя, несущего ответственность за выполняющийся процесс.

Исполнительный код используется для установки прав собственности на вновь создаваемые файлы, для проверки прав доступа к файлу и разрешения на посылку сигналов процессам через функцию *kill.*

Процессы могут изменять исполнительный код, запуская с помощью функции *exec* программу *setuid* или запуская функцию *setuid* в явном виде.

Программа *setuid* представляет собой исполняемый файл, имеющий в поле режима доступа установленный бит *setuid*. Когда процесс запускает программу *setuid* на выполнение, ядро записывает в поля, содержащие реальные коды идентификации, в таблице процессов и в пространстве процесса код идентификации владельца файла. Чтобы как-то различать эти поля, назовем одно из них, которое хранится в таблице процессов, сохраненным кодом идентификации пользователя. Рассмотрим пример, иллюстрирующий разницу в содержимом этих полей.

Синтаксис вызова системной функции *setuid*: *setuid (uid),* где uid - новый код идентификации пользователя.

В примере 6 приведен текст программы, демонстрирующей использование функции setuid. Предположим, что исполняемый файл, полученный в результате трансляции исходного текста программы, имеет владельца с именем "maury" (код идентификации 8319) и установленный бит setuid; право его исполнения предоставлено всем пользователям. Допустим также, что пользователи "mjb" (код идентификации 5088) и "maury" являются владельцами файлов с теми же именами, каждый из которых доступен только для чтения и только своему владельцу.

Пример 6.

```
#include <fcntl.h>
   main()
   {
      int uid,euid,fdmjb,fdmaury;
```

```
    uid = getuid();       /* получить реальный UID */
    euid = geteuid();     /* получить исполнительный UID */
    printf ("uid %d euid %d\n",uid,euid);
    fdmjb = open ("mjb",O_RDONLY);
    fdmaury = open ("maury",O_RDONLY);
    printf ("fdmjb %d fdmaury %d\n",fdmjb,fdmaury);
    setuid (uid);
    printf ("after setuid (%d): uid %d euid %d\n",uid,
        getuid (),geteuid ());
    fdmjb = open ("mjb",O_RDONLY);
    fdmaury = open ("maury",O_RDONLY);
    printf("fdmjb %d fdmaury %d\n",fdmjb,fdmaury);
    setuid (uid);
    printf ("after setuid(%d): uid %d euid %d\n",euid,
        getuid (),geteuid ());
}
```

В результате выполнения данной программы пользователю "*mjb*" выводится следующая информация:

uid 5088 euid 8319
fdmjb -1 fdmaury 3
after setuid(5088): uid 5088 euid 5088
fdmjb 4 fdmaury -1
after setuid(8319): uid 5088 euid 8319

Системные функции *getuid* и *geteuid* возвращают значения реального и исполнительного кодов идентификации пользователей процесса, для пользователя "*mjb*" это, соответственно, 5088 и 8319. Поэтому процесс не может открыть файл "*mjb*" (ибо он имеет исполнительный код идентификации пользователя (8319), не разрешающий производить чтение файла), но может открыть файл "*maury*". После вызова функции *setuid*, в результате выполнения которой в поле исполнительного кода идентификации пользователя ("*mjb*") заносится значение реального кода

идентификации, на печать выводятся значения и того, и другого кода идентификации пользователя "*mjb*": оба равны 5088. Теперь процесс может открыть файл "*mjb*", поскольку он исполняется под кодом идентификации пользователя, имеющего право на чтение из файла, но не может открыть файл "*maury*". Наконец, после занесения в поле исполнительного кода идентификации значения, сохраненного функцией *setuid* (8319), на печать снова выводятся значения 5088 и 8319.

3.3.6 Управление заданиями

Для просмотра списка текущих заданий используется команда **jobs.** В этом списке приводится порядковый номер задания, который можно использовать в любой команде, связанной с управлением заданиями. Использование ключей: **–p** позволяет выводить только идентификаторы групп процессов, **-l** – только идентификаторы заданий и групп заданий, **k** - информацию о заданиях, состояние которых изменилось со времени последнего запроса.

Для запуска команды в фоновом режиме необходимо добавить к ее имени символ амперсанта, например, *top &* .

Команда *logger>>file.log* позволит запустить программу ведения журнала logger , вывод программы запишется в файл file.log, сохраняя все предыдущие записи. Если файл не существует, то он будет создан.

3.4 Вопросы к защите лабораторной работы

1) Что понимается под термином “пользовательская среда UNIX”?
2) Напишите общий синтаксис скрипта.
3) Приведите примеры сложных синтаксических конструкций получения значений переменной.
4) Перечислите внутренние переменные shell, используемые в скриптах.

3.5 Литература

1) А. Робачевский Операционная система UNIX.-СПб.:BHV-Санкт-Петербург, 1997/2002- 528 с.