ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

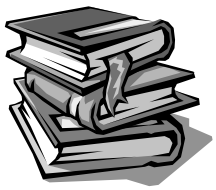
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
“МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРИБОРОСТРОЕНИЯ И ИНФОРМАТИКИ”

Кафедра “Персональные компьютеры и сети”

Р.Ф. Халабия

АДМИНИСТРИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Учебно-методическое пособие по выполнению лабораторных работ



Москва 2010

УДК 681.3.06(075.8)
ББК 32.973.202

Рекомендовано к изданию в качестве учебно-методического пособия редакционно-издательским советом МГУПИ

Рецензенты:

к.т.н., доцент Брейман А.Д.

Халабия Р.Ф. Администрирование вычислительных систем и сетей: Учебно-методическое пособие по выполнению лабораторных работ. – М.: МГУПИ, 2010. - 64 с.

Учебно-методическое пособие предназначено для студентов, обучающихся по направлению «Информатика и вычислительная техника» по специальности «Вычислительные машины, комплексы, системы и сети» по дисциплине «Администрирование вычислительных систем и сетей».

Лабораторные работы посвящены мониторингу локальных сетей в операционных системах семейства Unix, а также изучению проблем связанных настройкой и наладкой FTP и WEB серверов.

.

Содержание

Введение

Роль системного администратора компьютерной сети в жизни организации можно сравнить с ролью стоматолога. Несмотря на, как правило, весьма скромное вознаграждение (в государственных структурах), в современных компьютеризированных организациях он играет роль, важность и значение которой осознаются руководителями часто лишь при возникновении кризисных ситуаций, когда под угрозой оказывается финансовое положение предприятия, его имидж на рынке или репутация руководителя в глазах более высокого начальства. Тем не менее случись такая ситуация, руководитель постарается установить ее причины, и если будет доказана вина администратора сети, и без того скромное вознаграждение станет еще меньше. Но творчески и заинтересованно подходя к своей работе, вы не будете испытывать дискомфорта и ощущения "давления сверху". Сеть должна работать не потому, что этого требует руководство, а потому, что это СЕТЬ.

Администратор локальной сети — не столько профессия или должность, сколько образ жизни, а его квалификация — это жизненный опыт, опыт коллег, работающих в сетях, число которых ежегодно растет. С ростом числа сетей увеличивается и армия сетевых администраторов.

В данном учебно-методическом пособии даны описания некоторых полезных для работы администратора сети программ и утилит. Задачи по администрированию сети бывают настолько неожиданными, что для их решения в арсенале начинающего администратора не находится средств. Надеюсь, что время, потраченное на чтение и изучение примеров, не будет потеряно напрасно, и предложенный материал окажется полезным при выполнении ваших ежедневных задач, связанных с поддержанием надежности и работоспособности сети, ее развитием и совершенствованием, особенно в начале вашей работы.

1 Лабораторная работа №1. Мониторинг локальной сети на Flash-модели «Системный администратор»

1.1 Цель лабораторной работы

Цель работы – выявление и устранение неисправностей и сбоев оборудования в локальной сети при ее мониторинге.

1.2 Теоретические основы

Основной обязанностью системного администратора будет поддержка системы и обеспечение бесперебойной работы сервисов, а также поддержка работы оборудования в сетях. При работе с системой необходимо ее запускать и останавливать работу. При этом надо проверять, работает ли система, есть ли на дисках свободное место и корректно ли работают необходимые сервисы и оборудование. Очевидно, что как только поступит сообщение о том, что система перестала работать, руководство и пользователи сети сразу же обратят на это внимание.

В таких случаях необходимо знать технологии упреждающего мониторинга системы. Если правильно применять эти методы на практике, то можно, как правило, заранее знать о том, что система скоро может выйти из строя.

При мониторинге основное внимание нужно уделять четырем основным ресурсам: процессор, оперативная память, дисковая память, сеть. Расход этих ресурсов часто взаимосвязан, например, скорость доступа к диску зависит от кэширования, т.е. от оперативной памяти.

В сетевой среде для получения информации о других системах можно воспользоваться возможностями сети, например с помощью удаленного управления. Чтобы в каждой системе не регистрироваться и не запускать команды, целесообразно пользоваться локальными агентами. Тогда можно выполнять мониторинг какого-либо параметра системы и передавать на удаленную консоль отчеты о его состоянии.

Мониторинг периферийных устройств также необходим. Принтеры, устройства чтения компакт-дисков и другие устройства обычно хорошо работают в операционных системах, однако для этого надо выполнить соответствующие настройки.

Для мониторинга локальной сети и для устранения неполадок и сбоев оборудования в ней имеется набор утилит и команд, которые представлены в таблице 1.

Таблица 1 – Утилиты и команды

Утилиты и команды	Назначение и синтаксис
1	**2**
ping	Посылает пакеты ICMP ECHO_REQUEST хостам сети **СИНТАКСИС** **ping** [имя компьютера], [IP-адрес]
telnet	Осуществляет TCP соединение, используемое для передачи данных, с различной управляющей информацией. **СИНТАКСИС** **telnet** [имя компьютера], [IP-адрес]
man	Получение справочной информации. При запуске команды без аргументов она выдает формат своего вызова. **СИНТАКСИС** **man** имя_интересующей_вас_команды **man –k** ключевое_слово # список команд, относящихся к команде. Для того, чтобы срабатывала команда **man –k**, файлы документации должны быть предварительно проиндексированы. Достигается это командой **catman –w**
chmod	изменение режима доступа к файлам **СИНТАКСИС** **chmod** режим файл … При выполнении переустановить действующий идентификатор группы, если # есть 7, 5, 3 или 1; учитывать блокировку доступа, если # есть 6, 4, 2 или 0.

Продолжение табл. 1

1	2
ssh	Осуществляет вход на сервер или на рабочую станцию через удаленную машину, если там настроена такая служба. Служит в качестве замены командам rsh и rlogin. **СИНТАКСИС** **ssh** [-l имя_пользовательля] ИМЯ_ХОСТА [команда] Здесь ИМЯ_ПОЛЬЗОВАТЕЛЯ означает имя пользователя на удаленном хосте, с которым происходит соединение (задается операндом ИМЯ_ХОСТА). Операнд КОМАНДА, если он не опущен, указывает выполняемую на удаленном хосте команду. Команда **ssh** имеет еще целый ряд операндов. В операнде -c можно указать метод шифрования (idea/blowfish/des/3des/arcfour); в операнде -p указывается номер порта. Операнд -v рекомендуется использовать для получения информации о том, что происходит в процессе установления сеанса, например, при возникновении каких-либо проблем, в первую очередь задержках в соединении. Два операнда позволяют осуществлять перенаправление портов: **- L ПОРТ:ХОСТ:ПОРТХОСТА** позволяет перенаправить ПОРТ локального компьютера на ПОРТХОСТА удаленного компьютера, заданного в аргументе ХОСТ; **-R ПОРТ:ХОСТ:ПОРТХОСТА** осуществляет перенаправление ПОРТа на удаленном ХОСТе на порт локального хоста (последний аргумент). Конфигурационные файлы клиента включают общий файл /etc/ssh_config и личные пользовательские config-файлы. Последние перекрывают_ действие общего файла, и их, в свою очередь, можно заменить явным заданием операндов ssh. В конфигурационном файле задаются такие параметры, как методы шифрования, возможность перенаправления портов, разрешение на использование обычных механизмов rsh/rlogin при невозможности ssh-аутентификации, номер используемого на удаленном сервере порта и др.
grep	Поиск в файле по шаблону, заданному ограниченным регулярным выражением **СИНТАКСИС** **grep** [-b] [-c] [-i] [-l] [-n] [-s] [-v] ограниченное_регулярное_выражение [файл ...]
restart	Перезапустить систему
exit	Выход из системы

Продолжение табл. 1

1	2
ifconfig	Конфигурирование сетевых интерфейсов ядра. Она используется на этапе загрузки для настройки интерфейсов при необходимости. После этого она обычно используется только при отладке или настройке производительности системы. Если аргументы не переданы, **ifconfig** выдает информацию о состоянии активных интерфейсов. Если указан один аргумент интерфейс, выдается информация только о состоянии этого интерфейса; если указан один аргумент -a, выдается информация о состоянии всех интерфейсов, даже отключенных. Иначе команда конфигурирует указанный интерфейс. **СИНТАКСИС** **ifconfig** [interface] **ifconfig interface** [aftype] options | address ... Опции команды: **interface** - имя интерфейса. Обычно это имя драйвера, за которым идет номер устройства, например, eth0 для первого интерфейса Ethernet. **up -** помечает интерфейс как включенный. Это можно использовать для включения интерфейса после ifconfig down. Это происходит автоматически при установке первого адреса интерфейса. Если интерфейс был переустановлен при предыдущей пометке в качестве отключенного, аппаратное обеспечение будет переинициализировано. **down** - помечает интерфейс как отключенный. Когда интерфейс помечен как отключенный, система не пытается пересылать сообщения через этот интерфейс. При возможности, интерфейс будет переустановлен, чтобы отключить также прием. Это действие не отключает автоматически маршруты, использующие данный интерфейс. **arp** - включает использование протокола разрешения адреса (Address Resolution Protocol) при сопоставлении адресов на уровне сети и адресов на уровне связи (используется по умолчанию). В настоящее время это реализуется путем сопоставления адресов DARPA Internet и адресов Ethernet 10 Мбит/с. **-arp** - отключает использование протокола разрешения адреса (Address Resolution Protocol). **-promisc** - запрещает режим promiscuous. **allmulti** - включает или отключает режим all-multicast. В этом режиме все многоадресные (multicast) пакеты в сети будут приниматься этим интерфейсом. **-allmulti** - отключает режим all-multicast. **dstaddr addr** - устанавливает удаленный IP-адрес для двухточечной связи (например, по протоколу PPP). Это ключевое слово сейчас считается устаревшим; используйте вместо него ключевое слово pointopoint.

Продолжение табл. 1

1	2
	promisc - помещает интерфейс в состояние promiscuous. В широковещательной сети это заставляет интерфейс получать все пакеты независимо от того, были ли они предназначены для этой машины или нет. Это позволяет, используя фильтры пакетов, анализировать сетевой трафик. Обычно, это хорошая техника охоты на сетевые проблемы, которые иначе трудно отловить. Здесь весьма полезна утилита tcpdump. С другой стороны, это позволяет хакерам исследовать движение паролей по сети и делать другие черные дела. Одна защита против этого типа нападения: не позволять присоединяться к сети чужим компьютерам. Другой способ: использовать безопасные опознавательные протоколы, типа Kerberos, или SRA login. Эта опция соответствует флагу PROMISC. **metric N** - устанавливает стоимость маршрутизации для интерфейса равной n, вместо стандартного значения 0. Стоимость маршрутизации (routing metric) используется протоколом маршрутизации (см. routed). Большие стоимости делают маршрут менее предпочтительным; стоимости учитываются как дополнительные пересылки на пути к целей сети или хосту. **mtu N** - этот параметр устанавливает максимальный размер пакета (Maximum Transfer Unit — MTU) для интерфейса. Обычно нет необходимости менять значение этого параметра, но, в некоторых случаях, уменьшение значения MTU позволяет добиться устойчивой работы абонентов с очень низким уровнем сигнала. Кроме того, он может использоваться для изменения параметров туннельных интерфейсов. **netmask addr** - устанавливает маску сети IP для этого интерфейса. По умолчанию используется обычная маска сети класса A, B или C (что определяется по IP-адресу интерфейса), но можно установить любое значение. **add addr/prefixlen** - добавляет адрес IPv6 для интерфейса. **del addr/prefixlen** - удаляет адрес IPv6 для интерфейса. **tunnel aa.bb.cc.dd -** создает новое устройство SIT (IPv6-в-IPv4) — тоннель к указанной цели. **irq** - устанавливает аппаратное прерывание, используемое данным устройством. Не для всех устройств можно динамически менять значение IRQ. **io_addr addr** - устанавливает адрес начала области ввода-вывода для данного устройства. **mem_start addr** - устанавливает адрес начала области разделяемой памяти, используемой этим устройством. Это нужно лишь для немногих устройств. **media type** - устанавливает физический порт или тип носителя, используемый устройством. Не для всех устройств можно менять этот параметр, и для разных устройств могут поддерживаться различные значения. Типичные значения типа — 10base2 (коаксиальный кабель Ethernet), 10baseT (витая пара Ethernet 10 Мбит/сек), AUI (внешний передатчик) и т. д. Специальный тип носителя auto можно использовать, чтобы потребовать от драйвера автоматически определять тип носителя. Не все драйверы могут это делать.

Продолжение табл. 1

	broadcast [addr] - устанавливает широковещательный адрес. Широковещательный адрес обычно создается из сетевого адреса установкой всех бит части машины. Некоторые реализации IP используют другую схему, эта опция помогает приспособиться к этим странным средам. Если широковещательный (broadcast) адрес был установлен, ifconfig показывает флаг BROADCAST . **pointopoint [addr]** - зто ключевое слово включает двухточечный (point-to-point) режим интерфейса, означающий, что он обеспечивает непосредственную связь между двумя машинами, которую никто не прослушивает. Если указан также аргумент адрес, устанавливает соответствующий протоколу адрес другой стороны связи, как и устаревшее ключевое слово dstaddr. В противном случае, устанавливает или сбрасывает флаг IFF_POINTOPOINT для интерфейса. **-pointopoint [addr]** - это ключевое слово отключает двухточечный (point-to-point) режим интерфейса **hw class address** - устанавливает аппаратный адрес соответствующего интерфейса, если драйвер устройства поддерживает такую возможность. После ключевого слова hw необходимо указать имя класса оборудования, а также аппаратный адрес в текстовом виде. В настоящее время поддержимвается оборудование классов ether (Ethernet), ax25 (AMPR AX.25), ARCnet и netrom (AMPR NET/ROM). **multicast** - устанавливает у интерфейса флаг поддержки групповой передачи данных. Обычно в этом нет нужды, поскольку драйвер сам выставляет этот флаг. **address** - IP-адрес, присваиваемый интерфейсу. **txqueuelen length** - устанавливает длину очереди передачи для устройства. Это позволяет установить меньшие значения для более медленных устройств с продолжительными задержками (модемные линии, ISDN), чтобы быстрая передача больших объемов данных не слишком мешала передаче данных интерактивных сеансов, например, telnet.
apachectl	утилита управления Web-сервером (Apache) **СИНТАКСИС** **apachectl start** - запускает сервер Apache **apachectl stop** – останавливает сервер Apache **apachectl restart** – перезапуск сервер Apache **apachectl graceful** - перезапуск сервера Apache без прерывания имеющихся соединений

Окончание табл. 1

1	2
sudo	Запуск команды от имени другого пользователя. **СИНТАКСИС** **sudo** -K | -L | -V | -h | -k | -l | -v **sudo** [-HPSb] [-a auth_type] [-c class|-] [-p prompt] [-u username|#uid] {-e file [...] | -i | -s | command} sudoedit [-S] [-a auth_type] [-p prompt] [-u username|#uid] file [...] **sudo** позволяет разрешенным пользователям запускать команды, как суперпользователь или другой пользователь, определенный в файле sudoers. Настоящий и эффективный uid и gid устанавливаются в соответствии с используемым пользователем, который указан в файле passwd и вектор группы основан на файле группы (пока не будет использована опция -P). Если вызываемый пользователь root, или если целевой пользователь тот же, что и вызываемый, пароль не требуется. В иных случаях, sudo требует чтобы пользователи, аутентифицировали себя с паролем по умолчанию (ПРИМЕЧАНИЕ: в конфигурации по умолчанию, это пользовательский пароль, а не пароль root'a). Как только пользователь будет аутентифицирован, отметка времени обновится, а затем пользователь может использовать sudo без пароля, на короткий период времени (5 минут, если не изменено в файле sudoers).

1.3 Исходные данные для проведения лабораторной работы

Исходными данными для выполнения работы являются любой Интернет-браузер с настроенным Flash-проигрывателем, Flash-модель «Системный администратор» - main30.swf.

1.4 Порядок проведения лабораторной работы

1.4.1 Изучить теоретический материал лабораторной работы.

1.4.2 Запустить через браузер Flash-модель «Системный администратор» - main30.swf .

1.4.3 Устранить все неисправности и сбои локальной сети Flash-модели.

1.4.4 На основании полученной информации, составить отчет по лабораторной работе. Форма отчета представлена в приложении А.

2 Лабораторная работа №2. Анализ производительности системы ОС Solaris

2.1 Цель лабораторной работы

Ознакомиться со способами анализа производительности рабочих станций ОС Solaris, в том числе при помощи системы пейджинга.

2.2 Теоретические основы

Для оценки производительности системы мы можем задействовать несколько утилит, которые помогут выяснить, насколько загружены различные компоненты компьютера. Исходя из показателей загрузки и наших знаний о реальных потребностях запущенных процессов, мы сможем решить, изменить ли конфигурацию компьютера, настройки системы или же иначе распределить задачи между компьютерами в сети.

Прежде всего, следует выяснить, есть ли в системе узкие места. Например, постоянная загрузка процессора на 100% может говорить о том, что в системе запущено слишком много конкурирующих процессов, или о работе вычислительного процесса, который постоянно требует процессорное время (что вполне нормально, если только ваш сервер не используют хакеры для подбора или расшифровки паролей), или же о том, что мощности процессора недостаточно для решения задач возложенных на него, и этот компьютер требует модернизации (начальство против? - попробуйте распределить нагрузку в сети или поменять работу - что сейчас легче сделать?)

Если одновременно запущенные процессы мешают друг другу, непрерывно требуя процессорное время, увеличивая нагрузку на планировщик задач и отнимая время на постоянное переключение контекстов, следует подумать об их последовательном запуске - на однопроцессорном компьютере это может привести даже к увеличению скорости их выполнения. Если же при этом процессы конкурируют не только за процессорное время, но и за оперативную память, вызывая активный пейджинг, то их последовательный

запуск совершенно точно будет более эффективным, чем параллельный: задачи выполнятся быстрее и нагрузка на систему будет меньше.

Для оценки загрузки процессора и состояния памяти (достаточно ли оперативной памяти, не перегружена ли дисковая подсистема свопингом и т.п.) служат программы **top** и **sar**.

Программа **top** выводит информацию о процессах, которые наиболее активно занимают процессор.

Программа **sar** подобна **vmstat** и выдает статистику по работе системы, ее можно запустить для выдачи определенных параметров одномоментно или для периодического вывода сведений, например, для десятикратного измерения значений стандартного набора параметров с периодом 5 секунд.

Если необходимо увеличить размер конкретного раздела, то есть два пути: физически изменить размер раздела или создать метаустройство, которое физически будет состоять из нескольких разделов на одном или нескольких дисках, но система будет его считать одним логическим разделом. Второй путь напоминает создание Volume Set в системах Windows.

Чтобы физически изменить размер раздела, надо, чтобы на диске вслед за этим разделом было свободное пространство, еще не отданное ни одному разделу. Если там есть какой-то другой раздел, то его придется удалить, предварительно сохранив нужные данные из него. После этого потребуется выполнить резервное копирование всех данных увеличиваемого раздела в какой-то каталог другого раздела, удалить старый раздел, создать на его месте новый, больший, с помощью команды **newfs**, и затем восстановить файлы из резервной копии. Этот метод рекомендован для использования в любых системах UNIX. Однако он требует значительных затрат времени и дискового (или ленточного, в зависимости от того, где вы создаете резервную копию) пространства.

Второй способ годится только для Solaris (другие коммерческие системы UNIX имеют свои собственные средства решения этой проблемы, которые здесь не обсуждаются). Метаустройство создается командой **metainit**.

Программа **growfs**, которая служит для увеличения размера файловой системы, может модифицировать таблицу индексных дескрипторов и другие управляющие структуры так, чтобы можно было работать с увеличенной файловой системой без потери старых файлов. Увеличение возможно только после создания метаустройства, причем как для смонтированной, так и для несмонтированной файловой системы, в том числе даже во время работы других пользователей с этой файловой системой.

Синтаксис команды **growfs**:

/usr/sbin/growfs [-M точка_монтирования] [параметры_newfs] [rawdevice]

Аргументы команды **growfs** обозначают:

- точка_монтирования - точка монтирования файлововй системы, которую требуется расширить. При этом на время расширения произойдет блокировка файловой системы функцией **lockfs()**;
- параметры_newfs - те же параметры, которые может принимать программа **newfs** при создании новой файловой системы, см. описание **newfs**;
- rawdevice - имя файла прямого доступа для метаустройства в каталоге /dev/md/rdsk.

Команда **growfs** увеличивает размер файловой системы до размера указанного раздела.

Увеличение размера раздела выполняется посредством добавления нового раздела к метаустройству и последущего запуска **growfs**. При увеличении размера зеркала (т.е. уже существующего метаустройства с реализованным зеркалированием, или, иначе говоря, с RAID уровня 1) следует вначале увеличить каждую из частей зеркала с помощью **metaattach**, как показано ниже, а затем - всю файловую систему с помощью **growfs**.

Особым случаем является расширение журналируемого метаустройства (trans metadevice), которое состоит из двух устройств - главного и журналирующего. Увеличивается только размер главного устройства, а затем **growfs** "напускается" на само журналируемое метаустройство. Вообще говоря,

можно увеличить и размер журналирующего устройства, но это не является обязательным.

Програма **growfs** на время модификации файловой системы блокирует запись в нее. Можно сократить время блокировки файловой системы, выполняя ее увеличение по частям. Например, мы хотим увеличить файловую систему размером 2 Гбайт до размера 8 Гбайт. Можно это делать поэтапно, добавляя по 16 Мбайт за этап, дав ключ s для явного указания размера общего размера новой файловой системы на каждом этапе. Число, следующее за ключом s, интерпретируется как общее число секторов новой файловой системы на каждом этапе и должно быть кратно размеру цилиндра в секторах. Иначе говоря, файловая система должна содержать целое число цилиндров.

Представим себе, что требуется увеличить размер раздела /dev/dsk/c1t0d0s3, на котором расположена файловая система /export. Для этого нам потребуется вначале преобразовать этот раздел в метаустройство, поскольку добавлять дополнительное пространство можно только к метаустройству. Допустим, добавлять к существующему разделу мы будем пока еще пустой, не содержащий файловой системы раздел /dev/dsk/c2t0d0s3:

metainit -f d8 2 1 c1t0d0s3 1 c2t0d0s3

Эта команда вызывает объединение разделов /dev/dsk/c1t0d0s3 и /dev/dsk/c2t0d0s3 в новое метаустройство d8. Теперь изменяем /etc/vfstab так, чтобы файловая система /export монтировалась на метаустройство d8:

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	options
/dev/md/dsk/d8	/dev/md/dsk/d8	/export	ufs	2	yes	-

Демонтируем /export и снова монтируем его (при монтировании будет использовано новое устройство из /etc/vfstab):

umount /export

mount /export

Запускаем **growfs** для расширения файловой системы на новый раздел:

growfs -M /export/dev/md/rdsk/d8

Ключ M нужен программе **growfs** для того, чтобы можно было увеличить размер смонтированной файловой системы. В процессе изменения размера запись в файловую систему блокируется программой **growfs**.

Файл /etc/lvm/md.tab содержит таблицу метаустройств, которая служит файлом настроек для запуска программы **metainit** при старте системы.

Ограничения при работе с growfs

С помощью **growfs** можно расширять только файловые системы UFS (не важно, смонтированные они или несмонтированные). Расширенная файловая система не может быть уменьшена. Расширение файловой системы невозможно, если:

- на задействованном в ней устройстве находится файл учета запущенной системы acct, или

- включена система безопасности на уровне C2 и файл журналирования находится на расширяемом устройстве, или

- на ней находится локальный файл свопинга, или

- эта файловая система монтируется в каталог /usr или корневой каталог или является активным разделом свопинга.

Чтобы не искать номер индексного дескриптора по имени файла каждый раз, когда к файлу происходит обращение, система кэширует имена файлов и каталогов вместе с номерами их виртуальных индексных дескрипторов. Этот специальный кэш называется **кэш поиска имен каталогов** (directory name lookup cache - DNLC).

При открытии файла DNLC вычисляет соответствующий индексный дескриптор по имени этого файла. Если имя уже находится в кэше, то оно отыскивается очень быстро, поскольку не происходит сканирования каталогов на диске. Каждая запись DNLC в ранних версиях системы имела фиксированный размер, поэтому пространство для имени файла не могло превышать 30 символов, а более длинные имена файлов не кэшировались.

Начиная с Solaris 7, это ограничение снято. Если каталог содержит несколько тысяч записей, поиск номера индексного дескриптора конкретного файла отнимет много времени. В том случае, когда в системе открыто много файлов, а количество файлов в рабочих каталогах велико, высокая частота успешных обращений к DNLC очень важна.

Кэш имен каталогов не нуждается в настройке, хотя его размер зависит от значения maxusers. По умолчанию он определяется как (17xmaxusers)+90 (в Solaris 2.5.1) или 4x(maxusers + max_nprocs)+320 (в Solaris 2.6 и выше). Другие параметры, на которые тоже оказывает влияние maxusers, обсуждены в лекции 10.

Команда vmstat –s показывает частоту успешных попаданий в DNLC с момента начала работы системы.

Если частота промахов велика (попаданий обычно не должно быть меньше 90%), следует подумать об увеличении размера DNLC. Проверим этот показатель работы системы командой vmstat -s |grep 'name lookups'

Количество запросов к DNLC в секунду можно получить из поля namei/s в выводе команды **sar -a**:

А теперь запустим операцию, которая точно требует многократного обращения к DNLC: find / -name "top" &

sar -a 1 5

vmstat -s |grep 'name lookups'

При интенсивных файловых операциях и доступе ко многим каталогам одновременно эффективность кэша имен каталогов снижается, но по-прежнему остается достаточно большой.

При создании файловой системы для индексных дескрипторов выделяется отдельное пространство на разделе. Индексные дескрипторы хранят свойства файла, в том числе теневые индексные дескрипторы - расширенные права доступа. Количество индексных дескрипторов - это неизменяемая величина. Отсутствие свободных индексных дескрипторов в файловой системе

приводит к невозможности записывать файлы в те каталоги, которые расположены в этой файловой системе. Количество свободных индексных дескрипторов можно определить, проанализировав вывод команды **df -e**:

Так как размер кластера файловой системы, как правило, составляет 8 Кбайт, драйвер файловой системы при записи собирает данные в блоки по 8 Кбайт для большей эффективности использования дискового пространства. При записи файлов значительных размеров драйвер файловой системы группирует несколько блоков для того, чтобы вместо серии мелких операций ввода-вывода выполнить одну большую по объему операцию.

Количество группируемых блоков равно параметру файловой системы maxcontig . В версиях, предшествующих Solaris 8, этот параметр по умолчанию был равен семи, и, таким образом, операция ввода-вывода происходила одновременно с семью последовательными блоками и позволяла одновременно записать порцию данных размером 56 Кбайт. Такое значение по умолчанию связано с ограничениями конструкции в раннем оборудовании Sun: системы, основанные на архитектуре sun4, не способны передавать за одну операцию более 64 Кбайт. В архитектурах sun4c, sun4m, sun4d и sun4u таких ограничений нет. В Solaris 8 значение по умолчанию изменилось и составляет 16 блоков (128 Кбайт).

Изменение параметра maxcontig может серьезно повлиять на производительность файловой системы, при работе с которой преобладают последовательные операции по записи средних и больших (более нескольких десятков килобайт) файлов на диск. Если при доступе к файловой системе большинство операций ввода-вывода совершается с данными малого размера, то изменение maxcontig не принесет особой пользы.

Размер кластера файловой системы задается с помощью ключа **-C blocks** команды **newfs** при создании файловой системы. Для существующей файловой системы его можно изменить с помощью ключа **-a** команды **tunefs**.

Настройка maxcontig дает наилучший результат, если ее выполнить при создании файловой системы, так как это фактически определит размер кластера

файловой системы. Проведение настройки уже используемой файловой системы не даст столь же впечатляющего результата, поскольку блоки, которые уже были записаны, не будут перекомпонованы.

Чтобы оценить загрузку файловых систем, имеет смысл запустить команду iostat -x 5, в тот момент, когда интенсивность ввода-вывода будет наиболее близка к обычному (или пиковому, в зависимости от того, как вы хотите настроить файловую систему) уровню:

Средние значения загрузки можно получить из колонок kr/s и kw/s, в более ранних версиях системы для получения таких значений приходилось делить содержимое колонок "K/r" и "K/w" на "r/s" и "w/s" соответственно для получения средних показателей "прочитано килобайт в секунду" и "записано килобайт в секунду".

Для получения информации о распределении файлов по размеру можно использовать команду: quot -c file_system

Например, вот каково распределение в моей тестовой системе:

```
quot -c /export/home
...
1128    1       165156
1136    1       166292
1160    1       167452
1176    1       168628
…         ..          ……
1360    1       177324
1376    1       178700
1440    1       180140
…         ..          ……
1832    1       193516
2047    40      437396
...
```

Первая колонка вывода - это размер файла в блоках, вторая - число файлов такого размера, третья - общее число блоков, занятых файлами такого размера и меньшими.

Эффективное использование памяти и свопинга

Вопрос об установке в систему дополнительной оперативной памяти представляет собой классический вопрос выбора между ценой и

производительностью. Если цена важнее производительности, то при нехватке памяти увеличивают размер раздела свопинга, если важнее производительность - увеличивают объем оперативной памяти. Если же нет возможности сделать ни то ни другое, новые процессы не смогут быть запущены при нехватке виртуальной памяти (это можно заметить по сообщениям "Not enough space" или "WARNING: /tmp: File system full, swap space limit exceeded").

Если виртуальной памяти достаточно, но оперативной памяти хронически не хватает, система будет в значительной степени занята пейджингом и не сможет нормально откликаться на запросы. В такой ситуации можно наблюдать невиданную дисковую активность, а сканер страниц будет занимать до 80% времени процессора.

Индикаторами нехватки памяти в системе являются активность сканирования страниц на предмет пометки для выгрузки и активная работа устройства, на котором расположен swap-раздел.

При этом высокая активность может быть временной или постоянной. В любом случае, имеет смысл уяснить ее причину.

Частота сканирования страниц

Повышенная частота сканирования страниц (scan rate) - главный показатель того, что в системе перестало хватать оперативной памяти. Для просмотра значения scan rate используйте команды sar –g или vmstat.

При анализе частоты сканирования с помощью **vmstat** имеет смысл запустить эту программу с параметром 60 для получения статистики каждые 60 секунд:

Первую строку (суммарную статистику) можно игнорировать. Если показатель page/sr остается выше 200 страниц в секунду в течение длительного времени, это говорит о вероятной нехватке оперативной памяти в системе.

Постоянно низкое значение частоты сканирования страниц говорит о том, что системе хватает памяти. С другой стороны, высокое значение может быть вызвано активностью процесса (или нескольких процессов одновременно), читающего данные с диска или получающего их через сеть - эти данные не

только кэшируются операционной системой, но и занимают место в памяти процессов, поэтому вполне могут вызвать активный пейджинг. Таким образом, не следует слепо полагаться на обсуждаемые рекомендации и считать 200 страниц в секунду абсолютной догмой индикации нехватки памяти: действуйте по ситуации, смотрите, какие процессы в действительности запущены и насколько они требовательны к памяти.

Активность свопинга

Если устройство, на котором находится область свопинга, загружено вводом-выводом, это говорит о нехватке памяти. Можно оценить дисковую активность с помощью программы iostat. В Solaris 2.6 и выше следует использовать команду iostat –xPnce, для получения информации об активности передачи данных в/из конкретных разделов дисков, в Solaris 2.5.1 доступна команда iostat –xc, которая позволяет получить статистику ввода-вывода только для целого диска (физического устройства), без деления на разделы.

Если своп-раздел расположен на отдельном диске (что неплохо), версия системыне важна, в противном случае в старых системах оценить реальную загрузку диска передачами данных свопинга тяжелее.

Можно также использовать sar –d или vmstat.

Длинная очередь страниц для выгрузки свидетельствует о необходимости нарастить оперативную память в системе.

Использование памяти процессами

В системах Solaris, начиная с версии 2.6, есть возможность выяснить, какие программы сколько памяти занимают (и подробнее - размеры сегментов данных, кода и т.п.) с помощью программы **pmap**.

Для получения детальной информации дайте команду

/usr/proc/bin/pmap -x PID

Информация о размере процесса в оперативной памяти также содержится в колонке RSS вывода программ **top** и **ps** (используйте **ps -ly**).

В пакете SunPro есть отладчик dbx, который помогает находить источник утечки памяти в программе; для такой работы следует компилировать программу компилятором SunPro с ключом **-g**.

Статистику использования разделяемой памяти вы получите по команде

ipcs -mb

Эти программы следует использовать для определения размера процессов и основных потребителей памяти в системе.

Размер области свопинга очень важен для системы, так как недостаток виртуальной памяти приводит к тому, что не может стартовать новый процесс.

Для управления пространством свопинга (получения информации о нем, добавления и удаления разделов свопинга) используется программа **swap**. Получить информацию о текущем состоянии пространства свопинга можно с помощью **swap -l**.

Для выяснения общего объема виртуальной памяти, который включает в себя объем оперативной памяти и пространства свопинга вместе, следует использовать **swap -s** или **sar -r**.

Если своп-раздел смонтирован в **/tmp** как файловая система типа **tmpfs**, команда **df -k /tmp**, покажет общий объем свободной виртуальной памяти, включая оперативную память.

В Solaris применяются оба широко известных типа обмена страницами между оперативной памятью и пространством свопинга на диске: свопинг и пейджинг. Как мы уже знаем, пейджинг - это выгрузка тех страниц, которые давно не использовались, а свопинг - выгрузка всех страниц процесса. Свопинг в Solaris выполняется только при сильной нехватке памяти. Какой из двух способов освобождения оперативной памяти для текущих нужд использовать - свопинг или пейджинг, ядро решает, сопоставляя объем свободной оперативной памяти с ключевыми пар аметрами ядра. Эти параметры перечислены ниже.

physmem: общее количество страниц в оперативной памяти.

lotsfree: сканер страниц начинает работать, когда количество свободной оперативной памяти становится меньше lotsfree.

Значение по умолчанию - physmem/64, но оно может быть изменено в /etc/system. Сканер страниц по умолчанию запускается в режиме пейджинга. Частота сканирования (scan rate, столбик sr в выводе vmstat) устанавливается равной параметру slowscan , который по умолчанию равен fastscan/10.

minfree: пока объем свободной памяти находится между lotsfree и minfree, частота сканирования страниц растет линейно от slowscan к fastscan по мере уменьшения размера свободной памяти (рис. 1.1). Значение minfree по умолчанию - desfree/2, значение fastscan по умолчанию - physmem/4. Если свободной памяти становится меньше, чем desfree (что по умолчанию равно lotsfree/2), сканер страниц начинает запускаться с частотой 100 раз в секунду. За каждый свой запуск сканер страниц проверяет desscan страниц. Этот параметр изменяется динамически вместе с частотой сканирования.

maxpgio: этот параметр (в зависимости от конкретной аппаратуры он имеет значение 40 или 60) по умолчанию ограничивает частоту ввода-вывода на устройство пейджинга. Для современных дисков с частотой вращения больше 7200 оборотов в минуту можно установить значение maxpgio в сто раз больше количества жестких дисков, задействованных в свопинге.

throttlefree: когда свободной оперативной памяти становится меньше, чем определено в throttlefree (по умолчанию этот параметр равен minfree), запросы процессов на выделение им новых страниц памяти переводятся в состояние ожидания до тех пор, пока не появятся свободные страницы.

cachefree: имеет значение для систем Solaris 7 (или систем 2.5.1 и 2.6 с установленными самыми свежими обновлениями); если в этих системах параметр priority_paging установлен равным 1 (т.е. priority paging включен), то пока свободной памяти больше, чем lotsfree, освобождаются только страницы файлового кэша в памяти, а страницы процессов не затрагиваются. Значение по умолчанию - удвоенное lotsfree. Системы Solaris более поздних версий, начиная

с 8-й, имеют другой алгоритм освобождения оперативной памяти, и в них НЕ следует включать priority paging и устанавливать значение cachefree.

В системах Solaris до версии 7 включительно сканер страниц работает так: для выбранных сканером страниц обнуляется флаг "используемости" страницы, выбор страниц происходит со скоростью, которую можно посмотреть с помощью **vmstat** или **sar -g** (scan rate). После обработки handspreadpages страниц сканер проверяет, установлен ли флаг "используемости". Фактически, сканер состоит из двух процессов, один из которых идет по памяти и очищает флаги встреченных страниц, а второй следует за ним на некотором расстоянии и проверяет,не было ли новых обращений к этой странице (не установился ли снова этот флаг). Если флаг не установлен (обращений к странице за то время, пока сканер отмечал handspreadpages страниц, не произошло), то страница отправляется в своп. Параметр handspreadpages по умолчанию равен physmem/4.

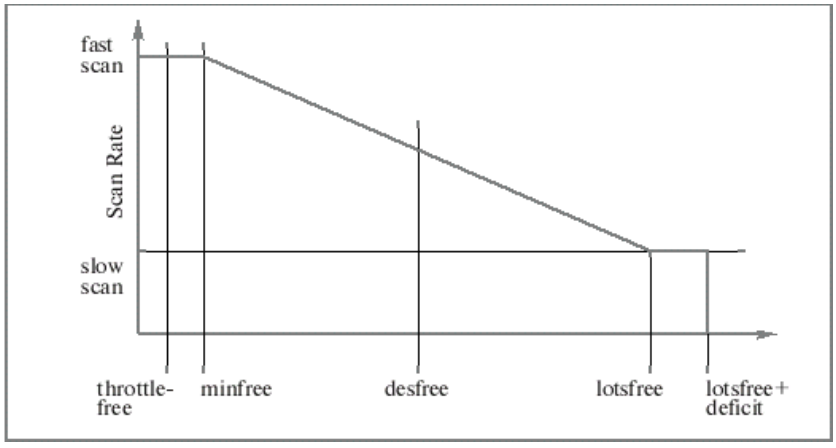


Рисунок 1.1 - Зависимость частоты запуска сканера страниц от объема свободной памяти

В системах Solaris 8 и более новых алгоритм освобождения памяти иной (он называется cyclical page cache). Он рассчитан на то, что при нехватке памяти выгружаются прежде всего страницы файлового кэша, и только затем - страницы процессов. Этот алгоритм разработан для тех же целей, что и priority paging в Solaris 7. Новый алгоритм использует два списка свободных страниц. Один - для помещения в него освобождающихся страниц файлового кэша,

другой - для помещения прочих освобождающихся страниц (разделяемой памяти, процессов и т.п.). При таком подходе файловый кэш ни с кем не соперничает за место в памяти.

В результате этих изменений в системах Solaris, начиная с версии 8, **vmstat** сообщает иные цифры, чем в той же ситуации в более старых системах, а именно:

- скорость возврата страниц выше;

- весь файловый кэш показывается как свободная память;

- скорость сканирования страниц (scan rate) низкая (даже близка к нулю), за исключением ситуаций, когда в системе ощущается острая нехватка памяти для приложений.

Для получения отдельного отчета по пейджингу страниц приложений (executables), данных (anonymous) и файловой системы используйте команду vmstat –p.

Если системе в течение некоторого времени (обычно 30 секунд подряд) не хватает памяти (объем свободной оперативной памяти падает ниже desfree), то начинается свопинг процессов. Планировщик задач выгружает те процессы, которые не претендовали на процессорное время в течение более чем maxslp секунд. По умолчанию maxslp равно 20. Этот режим свопинга называется мягким.

Если дело дошло до того, что памяти меньше, чем desfree, и, кроме того, два и более процессов выстроились в очередь к процессору, а активность пейджинга превышает maxpgio, начинается жесткий свопинг. Это означает, что ядро выгружает модули и страницы кэша, а затем начинает последовательно выгружать процессы до тех пор, пока объем свободной памяти не станет больше desfree.

Не выгружаются:

процессы классов планирования SYS и RT;

процессы, запускаемые в данный момент или остановленные по сигналу (например, при отладке);

процессы, завершающие работу;

процессы, находящиеся в состоянии зомби;

системные потоки;

процессы, которые блокируют другие, более высокоприоритетные потоки.

Значительный объем чтения и записи данных может вызвать нехватку памяти из-за стремления системы кэшировать весь ввод-вывод. При действительно существенных объемах ввода-вывода можно отменить кэширование и производить ввод-вывод напрямую.

Для этого можно использовать функцию directio() или параметр forcedirectio при монтировании файловой системы командой mount. Файловая система VxFS включает ввод-вывод в обход кэша всегда, когда объем операции ввода-вывода превышает значение параметра discovered_direct_iosz (см. man vxtunefs) (по умолчанию - 256 Кбайт).

Если в вашей системе преобладают множественные операции ввода-вывода небольших объемов данных и даже VxFS не помогает освободить память от большого количества кэшируемых данных, попробуйте уменьшить до приемлемого размера значение discovered_direct_iosz.

Если вы не уверены в том, что сетевой интерфейс вашего компьютера работает или, что он верно настроен, потребуйте эту информацию от ifconfig

Обратите внимание: псевдонимы интерфейсов и интерфейсы, не являющиеся сетевыми адаптерами Ethernet, не имеют MAC-адреса.

Сетевые интерфейсы, от которых ожидается работа в данный момент, должны присутствовать, иметь корректные IP-адреса, маски и верные MAC-адреса (00:00:00:00:00 в поле MAC-адреса вывода **ifconfig** говорит о том, что драйвер сетевого адаптера не считал адрес или не нашел адаптер). Кроме того, работающий интерфейс обязан находиться в состоянии UP.

Не забывайте давать команду ifconfig имя_устройства plumb, для активизации интерфейса, если интерфейс добавлен вручную. Эта команда создает необходимые драйверу сетевого адаптера потоки для работы с TCP/IP и

открывает доступ к устройству. До того, как будет дана эта команда, интерфейс не будет показан в выводе **ifconfig -a**, даже если драйвер и сетевой адаптер работают нормально.

Как узнать, работает ли ваша сеть? Попробуйте зайти по адресу www.playboy.com и сразу узнаете, есть ли доступ в Интернет. Лично я использую для проверки, есть ли связь с Сетью, менее одиозные сайты, например, домашнюю страницу главной финской сети FUNET. Это не так отвлекает от работы. В частности, потому, что на www.funet.fi почти все написано по-фински, а финский язык я знаю в объеме приветствий на вокзале. Может быть, стоит попробовать www.playboy.fi?

Проверка связи: ping

Для того чтобы выяснить, работает ли сеть в офисе, достаточно обратиться к соседнему компьютеру. А вдруг на нем нет web-сайта? Тогда на помощь приходит маленькая, но важная программа **ping.**

Вместо IP-адреса можно использовать имя компьютера.

Программа **ping** посылает маленький (обычно - 56 байт) пакет указанному компьютеру, а тот должен ответить таким же пакетом. Наш компьютер измеряет время прохождения пакетов туда и обратно и показывает его. Некоторые версии ping, в том числе и ping в Solaris 9, по умолчанию сообщают не время прохождения пакета, а сам факт ответа ("host is alive"). Если все же требуется время прохождения, вызывайте **ping** с ключом s:

Если **ping** показывает, что пакеты не проходят, это может случиться по нескольким причинам. Иногда **ping** выдает однозначную диагностику, а иногда приходится прерывать его, т.к. программа замирает в ожидании ответа, который никак не приходит.

Диагностика **host is down** говорит о том, что маршрутизация пакетов до искомого компьютера работает, а сам компьютер - нет. Возможно, его выключили, отключили от компьютерной сети или перевели в однопользовательский режим для профилактики.

Диагностика **no route to host** показывает, что маршрутизация пакетов до адресата не работает. Возможной причиной может быть сбой в таблице маршрутизации вашего компьютера (например, отсутствует запись об основном шлюзе).

Проверка соответствия адресов: arp

Если попытка "пинговать" тот или иной адрес в локальной сети дает странные результаты, может пригодиться программа **arp**, которая показывает локальную таблицу соответствий MAC-адресов и IP-адресов в сети:

Если сбой в **arp**-таблице произошел на компьютере, который является основным шлюзом для многих компьютеров в сети, то выход из сети может оказаться заблокированным для всех пакетов, которые пытаются выйти за пределы сети.

Проверка маршрутов**: traceroute** и **route.**

Чтобы проследить путь, по которому пакет данных движется к месту назначения, проходя по дороге через маршрутизаторы, используют программу **traceroute**.

Она прослеживает весь путь пакета, при этом по умолчанию инспектируются и показываются не более чем тридцать "hop'ов". Hop (читается "хоп") - это один переход от одного сетевого интерфейса до другого. Если говорят, что до получателя - один hop, это значит, что в дороге пакет не пройдет ни через один маршрутизатор, выполнит только один переход - от отправителя к получателю:

Идея **traceroute** такова: программа отправляет пакеты с адресом получателя, путь до которого мы хотим проследить. В поле TTL IP-пакета для начала ставится число 1. В поле номера порта ставится номер заведомо неиспользуемого порта (обычно это большое число). Естественно, первый же маршрутизатор на пути пакета сообщает, что TTL пакета истекло. Ответ маршрутизатора учитывается и выводится на экран, а затем посылается пакет с TTL, равным двум. Ответ про истечение времени присылает следующий маршрутизатор по пути следования пакета, и так происходит до тех пор, пока

пакет не дойдет до получателя. Когда это случится, сообщение об ошибке будет иным: "порт не обслуживается" (ведь порт специально задан таким, чтобы он не обслуживался).

Программа **traceroute** посылает три пакета с каждым из значений TTL, чтобы подсчитать среднее время прохождения пакета до каждого из промежуточных маршрутизаторов. Если ответ от какого-нибудь маршрутизатора не пришел за пять секунд, **traceroute** выводит звездочку (*) вместо времени ответа маршрутизатора, и это символизирует превышение тайм-аута.

Анализ маршрута пакета с помощью **traceroute** не всегда возможен, так как в некоторых сетях пересылка пакетов на нестандартные порты запрещена, а в некоторых - запрещена пересылка пакетов ICMP, в которые пакуются ответы маршрутизаторов типа "истекло TTL вашего пакета". Кроме того, пакеты разных типов могут проходить через маршрутизаторы по разным путям. Одинаковые пакеты могут быть направлены разными путями, в зависимости от загрузки сети. Поэтому маршрут пакета, показанный с помощью **traceroute**, верен только на момент выполнения этой программы. Вы можете применять **traceroute** и для проверки маршрутизации вашей сети. Например, если пакеты, предназначенные внешней сети, не отправляются на основной шлюз, а ищут другой путь (это будет видно посредством **traceroute** при попытке отследить путь пакета), то это верный признак сбоя в настройках.

Программа **traceroute** имеет ряд ключей для изменения параметров пересылаемых пакетов - смотрите **man traceroute**.

Для анализа и модификации локальной таблицы маршрутизации применяется программа **route**. Используйте программу **route** для добавления статических маршрутов и внесения оперативных изменений в маршрутизацию через ваш компьютер, когда это необходимо.

Безопасность в сети - это тема для отдельной книги или даже целого многотомника, поэтому здесь мы рассмотрим только основные аспекты, связанные с обеспечением безопасности систем Solaris в сети:

- все сетевые службы (демоны) в системе представляют собой корректно работающие программы, реализующие стандартные протоколы работы;
- система предоставляет доступ к ограниченному набору сетевых служб, этот доступ жестко контролируется посредством аутентификации и авторизации обращающихся, сообщение о разрешении доступа сохраняется в файле протокола;
- при работе в сети используются безопасные протоколы передачи данных, там, где это применимо, данные передаются зашифрованными надежными (на данный момент) алгоритмами шифрования;
- администратор(ы) и пользователи системы всегда применяют свои собственные пароли и никогда не сообщают их другим лицам.

Картина получилась идеальной. Если вы работаете в компании, где все происходит в точности так, как здесь описано, вы - счастливый человек!

Что надо сделать, чтобы приблизиться к идеалу?

Для того чтобы все программы работали так, как вы ожидаете, следует:

- регулярно следить за сообщениями о найденных в ПО уязвимостях. Для этого существует ряд списков рассылки, на которые вы можете подписаться совершенно бесплатно. Наиболее толковый вариант, по-моему, рассылка от CERT (Computer Emergency Response Team) - зайдите на www.cert.org и проверьте сами!
- всегда устанавливать рекомендованные рассылкой от CERT или производителем системы заплатки и обновления системы или отдельных программ;
- строго следовать рекомендациям производителей программ, например, если сказано, что демона squid нельзя запускать от имени привилегированного пользователя root, то и не надо этого делать!

Ограничение набора служб, к которым можно обратиться, так же как и ограничение доступа к каждой службе в отдельности, мы рассмотрим ниже, в разделе "Ограничение доступа к сетевым службам". О протоколировании работы демонов заботятся программисты, которые их пишут, поэтому

администратору надо знать лишь, где хранятся файлы протокола и к чему они относятся.

Применяйте только безопасные протоколы в сети, для этого:

- вместо telnet всегда используйте secure shell (ssh);
- используйте https вместо http там, где через web-интерфейс передаются конфиденциальные данные;
- запрещайте на сервере протоколы, позволяющие узнавать информацию о вашей сети (например, finger);
- пользуйтесь только теми программами, назначение которых вам известно.

Запретите всем пользователям системы (себе - тоже!) передавать важные пароли открытым текстом через сеть, сообщать кому-либо свой пароль в системе, реагировать на провокационные сообщения по e-mail (мошенники уже пытались убедить вас в том, что вам нужно где-то перерегистрироваться и сообщить им свой пароль?), оставлять свой пароль записанным на видном месте, придумывать примитивные пароли.

Список глупостей можно продолжить, но мы ограничимся призывом вести себя осторожно: от нас зависит безопасность сетей в большом числе организаций!

Сетевые службы делятся на два типа: запускаемые в начале работы системы и запускаемые по запросу. Те, что запускаются в начале, обычно функционируют постоянно, до завершения работы системы или до аварийного завершения. К таким службам относятся все демоны с относительно высокой постоянной нагрузкой, которые должны давать быстрый ответ клиенту: почтовые серверы, web-серверы, демоны **sshd** и многие другие. К запускаемым по запросу относятся службы, которые нужны реже, и между скоростью доступа к ним и эффективностью использования ресурсов часто выбирается эффективность (незачем отнимать ресурсы у системы, запуская постоянно действующие, но редко требуемые процессы). К службам второго типа относятся **telnetd**, **ftpd** и другие.

Службы по запросу запускает демон **inetd**. При запуске или при получении сигнала SIGHUP он читает файл конфигурации /etc/inetd.conf, где определено, какие службы можно запускать по запросу, и какие программы при этом следует запускать.

Демон **inetd** выполняет функцию привратника: как только пакет приходит к воротам системы, **inetd** определяет, какой процесс надо запустить, чтобы пакет смог добраться по назначению - к этому процессу. Программа **inetd** сверяет номер порта назначения в пакете с номером порта назначения в файле /etc/inetd.conf - там этот номер в мнемоническом виде указан в первой колонке. Для выяснения соответствия между номерами портов и их мнемоническими обозначениями служит файл /etc/services.

Для ограничения доступа к сетевым службам прежде всего следует отменить запуск всех служб, доступ к которым вы предоставлять не намерены. Для служб, запускаемых по запросу, это можно сделать, просто поставив знак комментария # (решетка) перед строкой, в которой указана соответствующая служба. Службы (демоны), запускаемые в начале работы системы, выключаются тоже довольно просто - достаточно удалить запускающий их скрипт из каталога /etc/rc?.d. Если они запускаются напрямую из /etc/inittab, закомментируйте соответствующую строку в этом файле.

После того, как мы гарантировали запуск только требуемых нам служб, приходит время определить специфические права доступа к этим службам. Мы можем ограничить доступ к ним из тех или иных источников. Можно сделать это посредством фильтра пакетов, как описано в лекции 13, либо внеся изменения в файл /etc/hosts.allow, в случае использования TCP wrapper - программы tcpd. Для включения механизма TCP wrapper при работе через inetd следует в файле /etc/default/inetd параметру ENABLE_TCPWRAPPERS присвоить значение YES (по умолчанию установлено NO, что означает "не использовать TCP wrapper>).

Производительность сети в большей степени зависит от используемого сетевого оборудования, чем от настроек системы. Фактически, некоторую

оптимизацию может дать изменение параметра MTU (maximum transmission unit) с помощью **ifconfig**, если трафик через сеть однороден и можно точно определить превалирующие размеры пакетов.

Однако с помощью ряда инструметов можно оценить, как идут дела в сети.

Используйте **netstat –i**, для получения статистики по интерфейсам системы:

Если **netstat** сообщает о большом количестве коллизий на интерфейсе, это может говорить о перегрузке сегмента сети. Вспомните, не генерирует ли какая-нибудь программа излишний или паразитный трафик, нельзя ли избежать передачи каких-то данных через сеть? Для тщательного изучения ситуации пригодится программа **tcpdump**, которая выводит на экран (перенаправьте вывод в файл для последующего анализа!) заголовки каждого пакета, проходящего мимо вашего сетевого интерфейса. Большое число коллизий также может говорить о том, что давно пора сменить старый дешевый концентратор (hub) на новый коммутатор (switch) - ведь сегодня коммутатор стоит дешевле, чем концентратор в свое время!

Некоторое число ошибок на интерфейсе допустимо, но если оно явно пропорционально трафику через интерфейс и превышает 1% от общего числа пересылок (это видно по выводу netstat), стоит изучить состояние кабелей. Может быть, на одном из них стоит стул? Или его жестоко зажали между стойками? Нет? Тогда наверное кто-то завязал на нем несколько узелков на память... Помните также, что плохо обжатые, небрежно подсоединенные или сильно запыленные вилки на кабеле, временные патч-корды, ставшие постоянными, также могут быть причиной проблем в сети. Значительное число ошибок на интерфейсе (10% и более) может сигнализировать о неисправности сетевого адаптера.

Бывает и так: сеть в полном порядке, **netstat** бодро сообщает, что ошибок и коллизий не имеется, новенькое сетевое оборудование цинично смотрит на нас матовым боком, а скорость передачи через сеть оставляет желать лучшего.

В этом, кроме оборудования, могут быть виноваты неверно настроенные драйверы (например, вы ожидаете **full-duplex** на интерфейсе, но сообщает ли вам о нем **ifconfig** или коммутатор, к которому подсоединен ваш быстрый компьютер?) или медленная дисковая подсистема получателя или отправителя данных.

2.3 Исходные данные для проведения лабораторной работы

Исходными данными для выполнения лабораторной работы являются критерии производительности системы и система пейджинга. Лабораторная работа проводиться под управлением операционной системы Solaris.

2.4 Порядок проведения лабораторной работы

2.4.1 Изучить теоретический материал лабораторной работы.

2.4.2 Проверить:

- Какие процессы занимают процессорное время.
- Частоту успешных попаданий в DNLC.
- Частоту промахов при многократном обращении к DNLC
- Коллизии в локальной сети.
- Объем виртуальной памяти.

2.4.3 Построить график статистики работы рабочей станции через каждые 60 сек.

2.4.4 Определить количество запросов к DNLC в секунду.

2.4.5 Оценить загрузку файловых систем.

2.4.6 Получить статистику использования разделяемой памяти.

2.4.7 Составить таблицу MAC-адресов и IP-адресов в локальной сети.

2.4.8 Найти файлы, где определено, какие службы можно запускать по запросу, и какие программы при этом следует запускать.

2.4.9 Оформить выполнение лабораторной работы в виде отчета. Форма отчета представлена в приложении Б.

3 Лабораторная работа №2. Оптимизация работы процессов в ОС Solaris

3.1 Цель лабораторной работы

Цель работы – изучить способы оптимизации работы процессов для увеличения производительности компьютера. Рассматриваются наиболее вероятные причины снижения скорости работы и способы их устранения.

3.2 Теоретические основы

Несмотря на кажущуюся простоту, компьютер состоит из большого числа частей, и быстродействие системы определяется тремя факторами: скоростью работы каждого компонента, согласованием быстродействия разных компонентов и точной настройкой системы для работы именно с этой конфигурацией аппаратуры. Здесь мы не будем касаться первых двух факторов и предположим, что мы имеем идеально согласованную конфигурацию аппаратных средств. Поскольку в реальной жизни такое встречается редко, наш глаз отдыхает при взгляде на этот прекрасный компьютер. Все, что мы можем сделать с ним - это установить подходящую операционную систему и настроить ее оптимальным образом.

Оптимизации подлежит следующее:

- количество одновременно запущенных процессов;
- количество потребляемой процессами памяти;
- объем оперативной памяти в системе;
- размер swap-раздела;
- набор ресурсов, в которых несколько процессов нуждаются одновременно.

Вопросы анализа и увеличения производительности дисковой подсистемы вы познакомились на выполнении лабораторной работы №1, сейчас мы затрагиваем только тему оптимизации работы процессов. Фактически, обсуждаются два вопроса: оптимизация использования памяти и

оптимизация приоритетов процессов. Первый, как легко догадаться, в практических задачах встречается много чаще.

Начнем с изучения того, как устроена виртуальная память в Solaris, ибо она является первым по значимости ресурсом, который постоянно делят между собой все процессы, запущенные в системе.

Поскольку процессам, запущенным в системе, обычно в сумме требуется больше места, чем допускает размер оперативной памяти, в любой системе UNIX предусмотрен механизм виртуальной памяти. Объем виртуальной памяти складывается из объема оперативной памяти и объема пространства свопинга (swap space). Подсистема виртуальной памяти в ядре заботится о том, чтобы с точки зрения процесса память была непрерывна и всегда доступна. В действительности страницы памяти, выделенные процессу, могут как угодно распределяться в оперативной памяти или быть выгруженными на диск в пространство свопинга.

Вся виртуальная память разбита на страницы объемом 4 Кбайт.

Некоторые компьютеры в силу их аппаратной реализации используют страницы памяти по 8 Кбайт. К ним относятся компьютеры с микропроцессорами DEC Alpha, первыми процессорами Sun SPARC (например, Ross RT601/Cypress CY7C601/Texas Instruments TMS390C601A, устанавливавшиеся в SPARCstation 2) и модели Sun UltraSPARC. В Solaris для определения фактического размера страницы памяти следует использовать программу /usr/bin/pagesize или функцию getpagesize(3C).

Потребителями виртуальной памяти в Solaris являются ядро системы, кэши файловой системы, тесно разделяемая память (intimately shared memory) и процессы. Тесно разделяемая память специфична для Solaris и представляет собой область разделяемой памяти, которую нельзя выгружать на диск. Тесно разделяемую память используют такие программы, как Oracle, Sybase, Informix.

Виртуальная память построена на четырех принципах, реализованных в системе.

Во-первых, каждый процесс получает отдельное **виртуальное адресное пространство** (virtual address space). Это значит, что процессу доступен определенный диапазон ячеек памяти. Максимальный размер этого диапазона памяти определяется длиной слова адреса в компьютере. Процесс, запущенный в 32-разрядной системе, будет иметь виртуальное адресное пространство размером 4 гигабайта (длина адреса - 32 бита). Подсистема виртуальной памяти соотносит (отображает) пользовательский кусочек виртуального адресного пространства и реальные страницы физической памяти.

Во-вторых, адресные пространства нескольких процессов могут перекрываться незаметно для процессов, если они используют общий код. Например, одновременно могут быть запущены три экземпляра одного и того же командного процессора (пусть это будет bash). Они имеют отдельные виртуальные адресные пространства. В каждом виртуальном пространстве находится экземпляр процесса командного интерпретатора, копия библиотеки libc и (возможно) копии других разделяемых процессами ресурсов. Подсистема виртуальной памяти незаметно для процессов отображает эти разделяемые куски памяти в одну и ту же область физической памяти так, что в физической памяти содержится всего один экземпляр разделяемого ресурса. Похоже на создание жестких ссылок на файл, верно?

В-третьих, подсистема виртуальной памяти выгружает наименее используемые страницы памяти на диск, когда физической памяти не хватает для всех процессов.

В-четвертых, подсистема виртуальной памяти запрещает процессу обращаться к ячейкам памяти из чужого адресного пространства, причем это делается на аппаратном уровне - посредством механизма диспетчеризации.

Для оценки памяти, занимаемой каждым из процессов, можно использовать как уже известные команды top и ps, так и команду pmap (последняя дает более подробное распределение памяти процесса по типам - разделяемая память и т.п.): pmap -x

Вообще говоря, в Solaris существует целое семейство так называемых процессных утилит (proc tools) или p-команд, работающих с файловой системой /proc, в которую отображаются многие структуры ядра, в частности, таблица процессов. Эти программы позволяют получать самую разную информацию о процессах, а некоторые из них могут также проанализировать завершившийся аварийно процесс, если от него остался файл core.

Не следует забывать, что память потребляется не только процессами, но и кэшем файловой системы, тесно разделяемой памятью и ядром! Если в системе не запускается СУБД Oracle или другое подобное приложение, скорее всего, тесно разделяемая память в системе не используется. В Solaris 8 и Solaris 9 для ядра и обязательно запускающихся системных приложений следует заранее предусмотреть не менее 32 Мбайт памяти и еще 16 Мбайт, если CDE тоже запускается. Рекомендованным для Solaris 9 объемом памяти (не считая память, которая требуется для специфических приложений - СУБД, почтового сервера и т.п.) считается 64 Мбайт, но оптимальным для системы, в которой работают с графическим интерфейсом, считается 128 Мбайт. Если планируется одноврменно запускать несколько ресурсоемких графических приложений, например, Mozilla и OpenOffice, следует, по крайней мере, удвоить этот рекомендованный объем.

Если пользователи обращаются только к нескольким сотням мегабайт данных, но делают это часто, то для кэширования всех этих данных должно хватать оперативной памяти. Это радикально ускорит работу.

Список свободных страниц (free list)

Список свободных страниц - это набор страниц, из которого страницы извлекаются по запросу процессов. Управление распределением памяти между процессами основано на этом списке. Процессы берут память из него и возвращают ее обратно по завершении. Сканер страниц также возвращает память в список свободных страниц так, как это описано в разделе "Алгоритм пейджинга".

Каждый раз, когда процесс запрашивает память, происходит так называемая страничная ошибка (page fault). Страничные ошибки делятся на три типа:

Легкая страничная ошибка (minor page fault) - процесс попытался получить доступ к странице, которая была изъята сканером страниц, но пока еще не использована повторно другим процессом.

Значительная страничная ошибка (major page fault) - процесс пытается получить доступ к странице, изъятой сканером страниц, которая использована повторно и в данный момент уже отдана другому процессу.

Ошибка копирования при записи (copy-on-write fault) - процесс пытается записать данные в страницу памяти, которая используется совместно с другими процессами.

Сейчас нам важны некоторые основные моменты, связанные с производительностью процессов.

После загрузки системы вся виртуальная память распределяется между процессами постранично. Кроме того, в ядре инициализируется специальная таблица, в которой хранятся состояния страниц. Несколько мегабайт памяти ядро резервирует для себя, а оставшееся пространство отходит списку свободных страниц. В какой-то момент, когда процесс запрашивает память, из списка свободных страниц извлекается одна страница, которая и поступает в распоряжение процесса. Такая схема, при которой память выдается по принципу "когда потребуется", называется **выделением страниц по запросу** (demand paging).

Если список свободных страниц уменьшается до размера lotsfree, ядро запускает специальный поток внутри себя - сканер страниц. Он начинает искать страницы, которые можно выгрузить на диск с тем, чтобы увеличить размер свободной памяти и пополнить список свободных страниц. Дабы не выгрузить страницы, к которым часто обращаются, сканер страниц работает по двухшаговому алгориму. Просматривая оперативную память в порядке возрастания адресов, он очищает бит MMU (бит "используемости") для каждой

страницы. Этот бит устанавливается, когда идет обращение к странице. Сканер страниц ведет просмотр далее, но через некоторое время проверяет бит используемости ранее просмотренных страниц, ожидая доступа к этим страницам и установки их битов используемости. Параметры slowscan и fastscan определяют то время, которое пройдет между очисткой бита MMU и его повторной проверкой, а именно:

- **slowscan** - первоначальная частота сканирования. При увеличении этого значения сканер страниц выполняет меньше ненужных заданий, но делает больше работы.

- **fastscan** - частота сканирования в ситуации, когда свободной памяти не осталось.

Если при повторном просмотре ссылочный бит какой-то страницы по-прежнему в исходном состоянии, это значит, что к данной странице не обращались.

Те страницы, чей бит "используемости" не был изменен в течение некоторого времени, выгружаются на диск, и освобожденная память пополняет список свободных страниц.

Некоторые страницы (например, принадлежащие разделяемым библиотекам) могут разделяться между многими процессами, и при записи в такую страницу возникает **ошибка копирования при записи (copy-on-write fault)**. Как только это произойдет, из списка свободных страниц извлекается чистая страница и создается копия первоначальной разделяемой страницы для того процесса, который требовал записать данные; в дальнейшем процесс работает именно со своей копией разделяемой страницы. Когда процесс завершается, все его страницы, за исключением тех, которые он делил с другими процессами, возвращаются в список свободных страниц.

Сейчас же мы должны представлять себе, что если программа vmstat сообщает о постоянной активности устройства свопинга, а частота сканирования страниц высока (в Solaris 8 и более новых версиях она вообще должна быть близка к нулю в обычной ситуации), то следует подумать об

уменьшении числа одновременно запущенных процессов или об увеличении объема оперативной памяти.

Всегда запускайте ровно столько демонов, сколько требуется. Например, если компьютер не является сервером NFS, не следует создавать файл /etc/dfs/dfstab, так как при его наличии автоматически запускается некоторое количество сетевых демонов. Мало того, что ненастроенные демоны могут дать злоумышленнику незапланированный доступ к компьютеру, так они еще и память занимают. Всегда используйте ps –ef, для контроля за количеством запущенных процессов. Не оставляйте без внимания запущенные процессы: если среди них есть незнакомый вам демон, стоит почитать man по нему, чтобы выяснить, нужен ли он в вашей конфигурации.

Некоторые программы, такие как web-сервер Apache или прокси-сервер squid, запускают несколько процессов, размножая самих себя или вспомогательные службы для увеличения производительности. По умолчанию количество запускаемых ими процессов сделано "средним", т.е. для слабо нагруженной системы оно слишком велико, а для перегруженной внешними запросами - слишком мало. Постарайтесь установить оптимальное значение - так вы сможете выиграть от нескольких мегабайт до нескольких десятков мегабайт памяти.

Проектом в Solaris называется единица администрирования, предназначенная для оптимального управления ресурсами системы. К проекту могут относиться любые пользователи и группы, и каждый пользователь или группа могут входить в несколько проектов. В большой системе удобно определить ряд проектов в базе проектов (файле /etc/project или соответствующем файле базы NIS).

Проект характеризуется уникальным идентификатором проекта (PROJID). Каждый пользователь обязательно относится к некоему проекту по умолчанию, и какой именно это проект, определяется при входе пользователя в систему. Пользователь обязательно имеет главный проект (по аналогии с главной группой), но может участвовать в нескольких проектах.

Каждый процесс также обязательно ассоциируется с каким-нибудь проектом. Это не обязательно главный проект пользователя, запустившего процесс, так как пользователь волен отнести запущенный им процесс к любому из проектов, участником которых он является. Отнести пользователя или группу к проекту можно либо в описании пользователя в файле /etc/user_attr, либо в файле проектов /etc/project. Для тех случаев, когда администратор не позаботился о том, чтобы отнести пользователей к определенным проектам, в системе имеется предопределенный проект default, к которому относятся все пользователи, группы и процессы, для которых явным образом не указано иное.

Главный проект пользователя определяется при входе в систему следующим образом:

- если в файле /etc/user_attr запись об этом пользователе имеет атрибут project, то в качестве главного проекта пользователю назначается указанный таким образом проект;

- если в /etc/project имется проект с именем user.UID, где UID совпадает с UID пользователя, то он назначается главным проектом пользователя;

- если в /etc/project есть проект group.groupname и groupname совпадает с именем главной группы пользователя, то этот проект назначается главным пользователю;

- если в базе проектов есть проект с именем default, то главным назначается он.

Проверка перечисленных условий производится в указанном выше порядке. В качестве базы данных проектов может использоваться не только файл /etc/project, но и база данных NIS или LDAP. Порядок обращения к службам имен (файлу, NIS или LDAP) определяется в файле /etc/nsswitch.conf:
project: files nis ldap

При использовании PAM может оказаться полезным также изучить страницу руководства pam_projects(5).

Если при входе для пользователя не удалось определить главный проект, вход пользователю запрещается.

При внесении изменений в базу данных проектов изменения коснутся только процессов, которые будут запущены после этого, и тех пользователей, которые войдут в систему после сохранения изменений. На уже запущенные процессы и уже работающих пользователей изменения не повлияют.

Файл /etc/project имеет следующий формат:

projname:projid:comment:user-list:group-list:attributes

где:

projname - это имя проекта (в нем не должно быть точек, запятых или двоеточий), то есть уникальный идентификатор проекта;

projid - неотрицательное целое число не большее 2147483647;

comment - описание проекта;

user-list - список пользователей, входящих в проект, имена через запятую;

group-list - список групп, входящих в проект, имена групп через запятую;

attributes - атрибуты проекта в формате имя=значение.

Везде, где указано "список", может стоять звездочка (подразумевает "все"), имя может быть предварено восклицательным знаком, что означает "кроме этого" (!groupname - все указанные группы, кроме groupname).

Помимо редактирования файла вручную вы можете пользоваться программами projadd, projmod и projdel для добавления, изменения или удаления проектов. Для получения информации о соответствии процессов проектам следует запускать программы ps, id, pgrep, prstat:

Синтаксис вызова pgrep:

pgrep -J projidlist

Программа выполняется как интерактивная на полном экране (подобно top).

В системе Solaris, начиная с версии 9 выпуска 12/03, появился демон укупорки ресурсов (Resource Capping Daemon), который управляет тем, как

процессы используют оперативную память. Управление выполняется на попроектной основе, т.е. ресурсы ограничиваются для конкретных проектов.

Демон укупорки ресурсов rcapd занимается ограничением потребления физической памяти для процессов, относящихся к проектам с установленными ограничениями. Существуют также программы rcapstat и rcapadm, предоставляющие возможность управления работой rcapd и получения статистики.

Настройка таблиц диспетчера памяти производится в три этапа:

1) вывод существующей таблицы в текстовый файл;

2) редактирование этого файла;

3) загрузка новой таблицы диспетчера в ядро.

Работа по выводу и загрузке таблиц осуществляется с помощью программы dispadmin.

Попробуем модифицировать таблицу диспетчера для класса разделения времени так, чтобы ни один процесс не получил приоритета 59 и ни один процесс не лишился этого приоритета, если мы его присвоим. Это может быть полезно в тех случаях, когда какие-то задачи надлежит вручную запускать с повышенным приоритетом. Конечно, это привнесет несправедливость в таблицу приоритетов нашей системы, и слепо следовать нашему тестовому примеру не стоит.

Посмотрим, как сейчас себя ведут наши процессы:

```
top
last pid: 825; load averages: 0.05, 0.11, 0.12 20:35:24
68 processes: 67 sleeping, 1 on cpu
CPU states: 99.8% idle, 0.2% user, 0.0% kernel, 0.0% iowait, 0.0% swap
Memory: 128M real, 12M free, 206M swap in use, 387M swap free
PID   USERNAME   LWP PRI   NICE   SIZE   RES    STATE   TIME   CPU    COMMAND
825   root       1   59    0      2260K  1340K  cpu     0:00   0.61%  top
345   root       1   59    0      57M    8648K  sleep   1:37   0.35%  Xsun
470   root       4   49    0      141M   55M    sleep   2:11   0.28%  soffice.bin
622   root       1   59    0      15M    2928K  sleep   0:02   0.03%  dtterm
461   root       1   49    0      15M    1864K  sleep   0:03   0.00%  dtterm
654   root       15  19    10     79M    10M    sleep   0:08   0.00%  java
434   root       5   59    0      22M    4060K  sleep   0:04   0.00%  dtwm
652   root       1   49    0      24M    3784K  sleep   0:01   0.00%  sdtimage
435   root       1   49    0      16M    1216K  sleep   0:00   0.00%  dtfile
672   root       1   49    0      4728K  740K   sleep   0:00   0.00%  bash
427   root       1   49    0      18M    0K     sleep   0:00   0.00%  dtsession
467   root       1   49    0      4728K  0K     sleep   0:00   0.00%  bash
650   root       1   49    0      3460K  0K     sleep   0:00   0.00%  more
```

```
649   root     1   49    0     3356K  0K     sleep   0:00   0.00%   sh
634   root     1   49    0     3304K  0K     sleep   0:00   0.00%   man
```

Теперь пусть приоритет 59 может получить только та программа, которой мы это разрешим, а все остальные по умолчанию не могут.

Для этого предварительно модифицируем таблицу диспетчера так, чтобы ни один процесс не получил приоритета 59. Вначале выведем текущую таблицу приоритетов:

dispadmin -c TS -g > prior

Теперь мы его изменяем так, как нам надо, и он становится иным. Загружаем этот файл, запустив **dispadmin -c TS -s prior.** Смотрим вывод top:

```
last pid:  836; load averages: 0.14, 0.14, 0.13  20:43:48
68 processes: 66 sleeping, 1 running, 1 on cpu
CPU states: 94.8% idle, 5.0% user, 0.2% kernel, 0.0% iowait, 0.0% swap
Memory: 128M real, 10M free, 204M swap in use, 389M swap free
PID   USERNAME   LWP   PRI   NICE   SIZE   RES   STATE   TIME   CPU     COMMAND
470   root       4     49    0      141M   57M   sleep   2:34   7.64%   soffice.bin
345   root       1     59    0      56M    7228K sleep   1:46   0.86%   Xsun
836   root       1     59    0      2260K  1336K cpu     0:00   0.77%   top
622   root       1     59    0      15M    2972K sleep   0:02   0.03%   dtterm
672   root       1     48    0      4728K  1176K sleep   0:00   0.03%   bash
654   root       15    49    0      79M    11M   run     0:08   0.02%   java
461   root       1     49    0      15M    1924K sleep   0:03   0.02%   dtterm
434   root       5     59    0      22M    4084K sleep   0:04   0.00%   dtwm
652   root       1     49    0      24M    3784K sleep   0:01   0.00%   sdtimage
435   root       1     49    0      16M    1216K sleep   0:00   0.00%   dtfile
427   root       1     49    0      18M    0K    sleep   0:00   0.00%   dtsession
467   root       1     49    0      4728K  0K    sleep   0:00   0.00%   bash
650   root       1     49    0      3460K  0K    sleep   0:00   0.00%   more
649   root       1     49    0      3356K  0K    sleep   0:00   0.00%   sh
634   root       1     49    0      3304K  0K    sleep   0:00   0.00%   man
```

Те процессы, которые по-прежнему имеют приоритет 59, можно перезапустить. Выявим их по команде ps -ecL | grep 59, остановим и перезапустим. Кроме того, можно регулировать приоритет процесса напрямую с помощью команды priocntl, как показано ниже. Понизим на 20 единиц приоритет всех процессов в классе разделения времени: priocntl -s -c TS -p -20. Снова запускаем top:

```
last pid:  987; load averages: 0.00, 0.03, 0.07 21:00:41
66 processes: 65 sleeping, 1 on cpu
CPU states: 99.4% idle, 0.0% user, 0.6% kernel, 0.0% iowait, 0.0% swap
Memory: 128M real, 6188K free, 202M swap in use, 391M swap free
PID   USERNAME   LWP   PRI   NICE   SIZE   RES   STATE   TIME   CPU   COMMAND
984   root       1     58    0      2260K  1336K cpu     0:00   0.11%  top
345   root       1     58    0      56M    7184K sleep   1:52   0.07%  Xsun
622   root       1     58    0      15M    3500K sleep   0:03   0.02%  dtterm
470   root       4     48    0      141M   57M   sleep   2:40   0.00%  soffice.bin
654   root       15    58    0      79M    13M   sleep   0:08   0.00%  java
434   root       5     58    0      22M    4360K sleep   0:04   0.00%  dtwm
461   root       1     58    0      15M    2628K sleep   0:03   0.00%  dtterm
652   root       1     58    0      24M    5312K sleep   0:01   0.00%  sdtimage
349   root       7     39    6      4532K  704K  sleep   0:00   0.00%  mibiisa
212   root       18    49    3      2872K  720K  sleep   0:00   0.00%  nscd
435   root       1     58    0      16M    1908K sleep   0:00   0.00%  dtfile
436   root       1     58    0      16M    1864K sleep   0:00   0.00%  sdtperfmeter
427   root       1     58    0      18M    1368K sleep   0:00   0.00%  dtsession
672   root       1     58    0      4732K  1196K sleep   0:00   0.00%  bash
276   root       1     58    0      2068K  580K  sleep   0:00   0.00%  xntpd
```

Настройку диспетчерских таблиц следует проводить с осторожностью. Неверные действия могут привести к потере устойчивости и неполадкам в работе производственной системы. Тестируйте внимательно!

Регулировать приоритет процесса, как показано выше, можно с помощью команды priocntl. Ключ -s означает требование установить приоритет. Ключ -p позволяет задать относительное изменение приоритета, а для указания конкретного признака процесса (идентификатора и т.п.) следует использовать ключ -i (признак идентификатора обозначается pid, другие признаки поименованы в руководстве по priocntl).

Например, для понижения приоритета процесса с PID, равным 200, используйте

priocntl -s -c TS -p -20 -i pid 200

Для вывода списка части процессов вместе с заголовком, используйте POSIX-совместимую программу grep: /usr/bin/ps -ecL |/usr/xpg4/bin/grep -E 'nscd|PID'

Алгоритм пейджинга и свопинга в Solaris предусматривает возможность явного указания границ свободной памяти в системе, по достижении которых вначале происходит активный пейджинг (выгрузка отдельных страниц), а при дальнейшем уменьшении свободной памяти - свопинг (выгрузка всех страниц процесса сразу).

3.3 Исходные данные для проведения лабораторной работы

Исходными данными для выполнения лабораторной работы являются тип процессов, выполняемых в системе, критерии производительности системы. Лабораторная работа проводиться под управлением операционной системы Solaris.

3.4 Порядок проведения лабораторной работы

3.4.1 Изучить теоретический материал лабораторной работы.

3.4.2 Проверить:

- Какие процессы занимают процессорное время.
- Службы имен.
- Какие демоны занимают оперативную память.
- Объем виртуальной памяти в системе.

3.4.3 Построить таблицу приоритетов в системе.

3.4.4 Оформить выполнение лабораторной работы в виде отчета. Форма отчета представлена в приложении Б.

4 Лабораторная работа №4. Построение FTP-сервера на основе операционной системы Linux

4.1 Цели лабораторной работы

Цель работы – изучение возможностей ОС Linux для построения FTP-сервера в локальной сети

4.2 Теоретический материал по теме лабораторной работы

Для настройки FTP-сервера в ОС Linux специально выделяют файлы конфигурации, где устанавливаются параметры и права доступа FTP-сервера.

Он расположен в каталоге */etc* и имеет имя *proftpp.conf*.

Рабочие файлы могут находиться в каталоге */home***.** Файлы логов должны храниться в папке */log*.

Для создания учетных записей пользователей и рабочих групп необходимо пользоваться командой *useradd.*

Для установки и изменения прав на файл или каталог осуществляется с помощью команды *chmod.* Числовое обозначение прав доступа определено следующим образом*:*

0 – прав нет

1- выполнение

2- запись

4- чтение.

Также могут понадобиться команды:

- вызов редактора – *mcedit*
- работа с FTP-сервером – *ftp*
- определение IP-адреса – *ifconfig*
- тестирование канала – *ping*
- запуск файлового проводника – *mc*
- помощь – *man [команда]*

Алгоритм настройки FTP-сервера следующий:

1) Установить пакет *proftpd* с помощью команды *sudo aptitude install proftpd*. Если FTP-сервер не будет использоваться постоянно, ответить на появившийся вопрос о способе запуска: *"самостоятельно"*.
2) Открыть файл /etc/shells командой *sudo nano /etc/shells*
3) Добавить в него строку */bin/false*
4) Создать в */HOME* каталоге папку *FTP-shared* командой *sudo mkdir /home/FTP-shared*
5) Создать пользователя с именем *userftp* команда *sudo useradd userftp -p parol -d /home/FTP-shared -s /bin/false* **вместо "parol" - ввести слово или фразу в качестве пароля**
6) В папке *FTP-shared* создать две вложенные папки: *sudo mkdir /home/FTP-shared/public sudo mkdir /home/FTP-shared/upload*
7) Присвоить нужные права созданным папкам командами *sudo chmod 755 /home/FTP-shared; sudo chmod 755 /home/FTP-shared/public; sudo chmod 777 /home/FTP-shared/upload*
8) Переименовать имеющийся конфигурационный файл *proftpd.conf* и создать новый: *sudo mv /etc/proftpd/proftpd.conf /etc/proftpd/proftpd.conf.old; sudo nano /etc/proftpd/proftpd.conf*
9) Добавить в него строки согласно вашего задания
10) После произведенных действий фтп-сервер будет иметь следующие параметры доступа: *user (пользователь): donet; password (пароль): parol (тот, что присвоен для userftp)*
11) Если нужно сделать анонимный доступ, следует закомментировать обе секции для donet и раскомментировать секцию для анонима
12) Сервер уже запущен, но с параметрами по умолчанию, перезапустить: *sudo /etc/init.d/proftpd restart*
13) Для проверки синтаксиса созданного конфиг-файла можно выполнить: *sudo proftpd -td5*

14) Что бы узнать, кто подключен к фтп-серверу в данный момент используется команда *ftptop* (клавиша t меняет отображение, q - выход), можно также использовать команду *ftpwho*

15) фтп-сервер с двумя папками, одна из них (public) доступна только на чтение, другая (upload) - на запись

Если нужно подключить какую-либо папку или партицию к FTP-серверу (например, проверить работу только что созданного FTP-сервера) без редактирования конфига пригодится команда: *sudo mount -o bind /здесь/путь/папки/что/я/хочу/расшарить/ /home/FTP-shared/public* или с доступом на запись: *sudo mount -o bind /здесь/путь/папки/что/я/хочу/расшарить/ /home/FTP-shared/upload.*

Таким образом, можно в срочном порядке временно подключить папку или диск и потом отмонтировать командой: *sudo umount /home/FTP-shared/public* или *sudo umount /home/FTP-shared/upload.* Для постоянного доступа к нужным папкам подключить их посредством fstab. Бэкап файла *fstab*: *sudo cp /etc/fstab /etc/fstab.old.* Открыть файл */etc/fstab* комадой *sudo nano /etc/fstab* и добавить нужные пути: */здесь/путь/папки/что/я/хочу/расшарить /home/FTP-shared/public none bind 0 0.*

Теперь даже при рестарте сервера (компьютера) информация будет доступна, если сервер за роутером то только в локальной сети,. Что бы увидеть фтп-сервер из интернета нужно дать ему внешний ip-адрес. Для этого следует открыть нужный порт (в данном случае 21) для локального адреса (вида 192.168.xxx.xxx) на котором висит сервер, для доступа извне.

Следующим шагом нужно дать внешнему динамическому IP-адресу осмысленный и постоянный адрес. Сделать это можно при помощи сервиса DynDNS.com, создав при помощи его удобный и запоминающийся адрес (вида moj-server.homeip.net). Внести регистрационные данные с сервиса DynDNS в настройки роутера и поменять ServerName "server" в файле *proftpd.conf* на

ServerName "moj-server.homeip.net". Рестарт FTP-сервера : *sudo /etc/init.d/proftpd restart*

4.4 Исходные данные для проведения лабораторной работы

Исходными данными для выполнения лабораторной работы являются учетная запись, IP-адрес FTP-клиента. Лабораторная работа проводиться под управлением операционной системы Linux.

4.5 Порядок проведения лабораторной работы

4.5.1 Изучить теоретический материал лабораторной работы.

4.5.2 Настроить FTP-сервер со следующими параметрами:

- Порт – *8021*
- Максимальным количеством пользователей – *2*
- Учетная запись – *student* и пароль – *ftppass*
- Рабочая группа – *ftpgroup*
- Без возможности доступа анонимных пользователей
- Имя сервера – *Фамилия студента*

4.5.3 Создать простой файл и переслать на компьютер соседа и продемонстрировать работу настроенного сервера преподавателю.

4.5.4 Оформить выполнение лабораторной работы в виде отчета. Форма отчета представлена в приложении А.

5 Лабораторная работа №5. Построение Web-сервера на основе операционной системы Linux

5.1 Цели лабораторной работы

Цель работы – изучение возможностей ОС Linux для построения Web-сервера в локальной сети

5.2 Теоретический материал по теме лабораторной работы

В операционной системе Windows установка и настройка WEB-сервера не вызывает никаких сложностей... Впрочем, как и при использовании операционной системы Linux.

Рано или поздно любой WEB-разработчик откажется от использования операционной системы Windows и перейдет на Linux. В первую очередь давайте разберем, что же на самом деле такое *LAMP*. *LAMP* - подборка или комплекс специализированного программного обеспечения. *LAMP* включает в себя:

- Linux - Unix подобная операционная система;
- Apache - WEB-сервер;
- MySQL - База данных;
- PHP - интерпретатор языка высокого уровня PHP.

Так уж исторически сложилось, что эти компоненты стали связкой, так как изначально они не разрабатывались для работы друг с другом, а шли как отдельные проекты, но их свободное распространение и качество позволили использовать их вместе. Это и есть стандартный набор программного обеспечения под аббревиатурой *LAMP*.

Алгоритм установки и настройки Web-сервера.

1) Установить **Apache** командой *sudo apt-get install apache2.*
2) *Те*стируем его работоспособность, перейдя по ссылке *http://localhost/,* либо *http://127.0.0.1*. Если вдруг что-то пошло не так и страница не

открылась - попробуйте перезапустить Apache командой: *sudo /etc/init.d/apache2 restart*

3) Устанавить **PHP** командой *sudo apt-get install php5 libapache2-mod-php5*
4) Перезапускаем **Apache**: *sudo /etc/init.d/apache2 restart*
5) Проверить PHP на работоспособность. Если что-то пошло не так, то надо найти причины.
6) Создать файл в локальной директории сервера для **PHP** скрипта: *sudo gedit /var/www/test.php*
7) В открытый редактором файл, списываем следующий **PHP** код:

```
<?php
phpinfo();
?>
```

8) Сохранить изменения в этом файле. Далее для просмотра результата скрипта перейти по ссылке *http://localhost/test.php*. В результате вы должны увидеть таблицу с настройками **PHP** интерпретатора. После просмотра удалить файл, чтобы никто не смог просмотреть вашу конфигурацию: *sudo rm /var/www/testphp.php.*

9) Устанавить **MySQL** командой *sudo apt-get install mysql-server*

При установке MySQL программа попросит вас сконфигурировать систему. Необходимо создать пользователя (по-умолчанию root) и присвоить ему пароль. Обязательно указать пароль, чтобы никто иной не смог воспользоваться вашей Базой Данных.

10) После установки нам будет необходимо сконфигурировать следующие файлы:

- my.cnf - Конфигурационный файл MySQL;
- php.ini - Конфигурационный файл интерпретатора PHP;
- ... sites-available/default - Список и настройки виртуальных хостов Apache;

11) Если необходима возможность подключения к вашей БД, MySQL с удаленного компьютера отредактируем файл my.cnf: *gksudo gedit /etc/mysql/my.cnf*

12) В открывшемся файле необходимо закомментировать строку *bind-address = 127.0.0.1.* Строка должна выглядеть так: *# bind-address = 127.0.0.1*

13) Сохранить документ.

14) Чтобы База Данных MySQL работа с интерпретатором PHP для поддержки работы скриптов с MySQL - отредактировать файл php.ini: *gksudo gedit /etc/php5/apache2/php.ini*

15) В это файле найти строку *# extension=mysql.so*, чтобы строка выглядела так:

;extension = mysql.so

16) Сохранить документ и перезагрузить **Apache**: sudo /etc/init.d/apache2 restart

17) По-умолчанию в Web-сервере **Apache** отключена возможность использовать такой полезный файл как *.htaccess*. Включим его, так как он необходим для работы большинства скриптов и CMS.

18) Откройте файл: *gksudo gedit /etc/apache2/sites-available/default*

19) В этом файле вы должны изменить все строки (их там около трех) *«AllowOverride None»* на *«AllowOverride All»*. Затем сохранить документ.

20) Устанавить PhpMyAdmin (Web-интерфейс для работы с Базой Данных MySQL) *sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin*

21) Установка не требует вашего вмешательства и настройки. Единственное это то, что должны сделать ярлык на этот скрипт, так как по-умолчанию он устанавливается не в локальную директорию сервера, по-этому: *sudo ln -s /usr/share/phpmyadmin/ /var/www/pma*

22) Теперь данный интерфейс доступен по ссылке: *http://localhost/pma*

5.4 Исходные данные для проведения лабораторной работы

Исходными данными для выполнения лабораторной работы являются учетная запись, IP-адрес Web-клиента. Лабораторная работа проводиться под управлением операционной системы Linux.

5.5 Порядок проведения лабораторной работы

5.5.1 Требуется настроить WEB-сервер со следующими параметрами:

- MySQL
- PHP
- PhpMyAdmin
- Имя сервера – *Фамилия студента*
- IP-адрес вашего персонального компьютера

5.5.2 Создать тестовые примеры для проверки работы WEB-сервера и продемонстрировать работу настроенного сервера преподавателю.

5.5.3 Оформить выполнение лабораторной работы в виде отчета. Форма отчета представлена в приложении Б.

Вопросы для самопроверки

1. Что такое мониторинг локальной сети?
2. Какие параметры локальной сети необходимо проверять при мониторинге?
3. Что необходимо уметь и знать при устранении сбоев локальной сети?
4. Какие процессы влияют на производительность системы в целом?
5. Какими командами можно проверить состояние процессов, влияющих на производительности системы?
6. Как влияет размер виртуальной памяти на производительность системы?
7. Какие параметры влияют на производительность системы в целом?
8. Какими командами можно проверить состояние производительности системы?
9. Чем занимается система пейджинга?
10. Что такое порт?
11. Как задается учетная запись?
12. Каким образом устанавливаются права на файлы в FTP-сервере?
13. Назначение программы Apache?
14. За что отвечает конфигурационный файл web-сервера httpd.conf?
15. Назовите самый простой способ защиты непубличных Web-каталогов?

Список рекомендуемой литературы

1. Адельштайн Т., Любанович Б. Системное администрирование в Linux. – СПб.: Питер, 2010. – 288 с.

2. Клейменов С. А. Администрирование в информационных системах: учеб. пособие для студ. высш. учеб. заведений / С. А. Клейменов, В. П. Мельников, А. М. Петраков ; под ред. В. П. Мельникова. — М.: Издательский центр «Академия», 2008. – 272 с.

3. Торчинский Ф.И. Операционная система Solaris. Курс лекций: Учебное пособие. – М.: ИНТУИТ.ру, 2006. - 472 с.

4. Richard McDougall, Jim Mauro, Brendan Gregg. Solaris Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris. – Network: Prentice Hall, 2006. – 496 с.

Приложение А.

Бланк отчета по лабораторной работе №1

Студент _____

Шифр_____

Лабораторная работа №1. Мониторинг локальной сети на Flash-модели «Системный администратор»

ОТЧЕТ

1. **Дать определения мониторинга сети**

_____
_____
_____

2. **Таблица сбоев и**

№	Название сбоя	Устранение сбоя (как нашли причину сбоя и он устранен)
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

Дата сдачи _____ Подпись преподавателя_____

Приложение Б.

Бланк отчета по лабораторной работе №2

Студент _____

Шифр_____

Лабораторная работа №2. Анализ производительности системы ОС Solaris

ОТЧЕТ

1. **Что влияет на производительность системы (перечислите факторы)**

_____
_____
_____

2. Заполните таблицу

Частота успешных попаданий в DNLC	
Частоту промахов при многократном обращении к DNLC	
Коллизии в локальной сети	
Какие процессы занимают процессорное время	
Объем виртуальной памяти	
количество запросов к DNLC в секунду	
файлы, где определено, какие службы можно запускать по запросу, и какие программы при этом следует запускать	

2. Таблица MAC-адресов и IP-адресов в локальной сети.

MAC-адрес	**IP-адрес**	**Имя компьютера**

3. График статистики работы рабочей станции через каждые 60 сек.

4. График статистики использования разделяемой памяти

5. Загрузка файловых систем.

Устройство	**Имя**	**Его загрузка**

Дата сдачи _____ Подпись преподавателя_____

Приложение В.

Бланк отчета по лабораторной работе №3

Студент _____

Шифр_____

3 Лабораторная работа №2. Оптимизация работы процессов в ОС Solaris

ОТЧЕТ

1. **Что влияет на производительность системы (перечислите факторы)**

_____

_____

_____

2. Заполните таблицу

Частота успешных попаданий в DNLC	
Частоту промахов при многократном обращении к DNLC	
Коллизии в локальной сети	
Какие процессы занимают процессорное время	
Объем виртуальной памяти	
количество запросов к DNLC в секунду	
файлы, где определено, какие службы можно запускать по запросу, и какие программы при этом следует запускать	
Какие демоны занимают оперативную память	
Службы имен	

3. Построить таблицу приоритетов в системе.

Приоритет	Имя процесса	Меняется или нет

Дата сдачи _____ Подпись преподавателя_____

Приложение Г.

Бланк отчета по лабораторной работе №4 и №5

Студент _____

Шифр_____

Лабораторная работа № __«_____»

ОТЧЕТ

1. **Исходные данные для выполнения работы**

2. **Перечислите файлы, где вы меняли конфигурации. Выписать строчки, в которые вы изменили в каждом файле с комментариями**

3. **Перечислите команды, которыми Вы воспользовались**

Дата сдачи _____ Подпись преподавателя_____

Приложение Д

Пример оформления титульного листа отчета

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРИБОРОСТРОЕНИЯ И ИНФОРМАТИКИ**»

Кафедра ИТ-4 «Персональные компьютеры и сети»

ОТЧЕТ

по выполнению лабораторных работ

по курсу **Администрирование вычислительных систем и сетей**

Студент Сидоров Алексей Петрович Шифр 09132

Курс 5 Группа ИТ4-0801 Бюджет

ИТ4-0802 Платная

ИТ4-0801З Заочное

Форма обучения дневная (заочная)

Преподаватель _____ /Р.Ф. Халабия/

Москва 2010

Учебное издание

Халабия Рустам Фарук

АДМИНИСТРИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Учебно-методическое пособие по выполнению лабораторных работ

ЛР № 020418 от 08 октября 1997 г.
Подписано в печать _____
Формат 60x84 1/16
Объем 4,0 п.л. Тираж 100 экз. Заказ № _____

ГОУ ВПО "Московский государственный университет приборостроения и информатики"
107996, Москва, ул. Стромынка, 20