ADMINISTRAÇÃO DE SISTEMAS

UNIX

1. INTRODUÇÃO

1.1. *AMBIENTE UNIX DISTRIBUÍDO*

- **Cada estação de trabalho possui sua própria CPU, memória e sistema de I/O**
- **Procedimentos de *backup* envolvem várias máquinas e vários discos**
- **Cada estação de trabalho possui o seu próprio superusuário**
- **Utilitários tais como NFS e NIS tornam a administração diferente e mais fácil.**

NOTAS:

1.2. *TIPOS DE EQUIPAMENTO*

- **Standalone**
- **Servidor**
- **Diskless**
- **Dataless**
- **X-terminal**

NOTAS:

Uma estação de trabalho *standalone* possui uma área de root completa mais o diretório /usr em seu disco. Não necessita de serviços da rede para funcionar e faz o boot a partir de seu próprio disco. Pode ou não possuir uma unidade de fita acoplada.

Um servidor se caracteriza pela quantidade substancial de espaço em disco para uso de outras máquinas da rede, chamadas clientes. Normalmente possui uma unidade de fita.

Uma máquina *diskless* é aquela que necessita dos serviços da rede para acessar o disco de um servidor. Ela recebe o código para boot através da rede. Normalmente as áreas root e swap de uma máquina *diskless* residem em um mesmo servidor, mas o código executável e os arquivos dos usuários podem residir em várias máquinas da rede.

Uma máquina *dataless* é aquela que possui seu próprio disco para armazenar a área de swap e a área de root. Este disco pode também ser usado para armazenar dados de usuários. Arquivos e código executável necessários para sua operação (/usr) são obtidos da rede. Desta forma o código executável de diversas máquinas *dataless* é armazenado em um único servidor.

1.3. *TAREFAS DO ADMINISTRADOR*

- **Instalação de Software**
  - **Sistema operacional e atualizações**
  - **Aplicações de correções**
  - **Formatação e particionamento de discos**
- **Customização do software**
  - **Kernel**
  - **Terminais**
  - **Modens**
  - **Impressoras**
  - **Rede**
- **Segurança**
  - **Adição/remoção de usuários**
  - **Criação de grupos de usuários**
  - **Controle de acesso às máquinas**
  - **Restringir acesso privilegiado às máquinas**

NOTAS:

A administração de sistemas envolve a tarefa de sempre manter o sistema operacional atualizado, rodando a última versão disponível. Quando novas máquinas são adquiridas estas devem ser configuradas, o sistema operacional instalado e conectadas à rede. Esta instalação vai depender da configuração da máquina, se *diskless*, *dataless*, *standalone* ou servidora. É recomendável sempre formatar o disco antes de instalar o sistema operacional.

O **kernel** que vem com a máquina raramente atende às necessidades do usuário. Ele deve ser configurado e adaptado às condições do ambiente operacional. Hardware acrescentado à configuração básica normalmente requer a geração de um novo **kernel** para inclusão dos *drivers* de dispositivo.

A segurança é um aspecto que nunca deve ser negligenciado. Os os usuários devem ser gerenciados de forma a só poderem fazer aquilo a que forem designados. Enfatizar a importância do sigilo da senha de acesso e restringir o acesso privilegiado às operações do sistema. Criar grupos de usuários de modo que , pessoas trabalhando em um mesmo projeto possam compartilhar seus dados. Ao mesmo tempo impede o acesso de usuários não pertencentes ao grupo.

- **Serviços de rede**
  - **NFS**
  - **NIS**
- **Manutenção do Sistema**
  - **Manutenção do sistema de arquivos**
  - **Backups**
  - **Restore**
- **Estabelecer comunicação**
  - **E-mail**
  - **Usenet**
  - **Servidor de FTP anonymous**
  - **WWW**

NOTAS:

Um dos recursos mais importantes que o sistema Unix oferece é a capacidade de troca de mensagens entre usuário.

NFS (*Network File System*) permite que máquinas montem diretórios residentes em outras máquinas da rede. O acesso a estes diretórios é feito de maneira transparente para o usuário que tem a impressão de estar usando arquivos locais. NIS (*Network Information Services*) permite que informações importantes de configuração e outras sejam compartilhadas por diferentes máquinas.

A tarefa mais importante de um administrador de sistemas é fazer *backups*. Inevitavelmente dados serão perdidos, defeitos em disco ocorrerão. Backups devem ser realizados de forma sistemática e regular. Os comandos ufs**dump** e ufs**restore** existem para facilitar essa tarefa. O comando **fsck** é usado automaticamente para verificar a consistência do sistema de arquivos durante o boot do sistema e reduzir a possibilidade de um sistema de arquivos se corromper.

1.4. *ROOT*

- **Conta do superusuário**
- **Cuidados especiais com a password**
- **Nunca logar diretamente como root**

NOTAS:

A administração do sistema é função da conta root. Root é o nome do usuário mais privilegiado do sistema, também conhecido como superusuário. Esta conta pode fazer qualquer coisa no sistema. Como de praxe em sistemas Unix, nenhum ato do superusuário é questionado. Devido a este poder, é de extrema importância que a senha da área root seja protegida com muito cuidado. Ela não deve ser facilmente deduzida (nome da máquina, nome invertido do administrador, iniciais, nome do cachorro, nome do carro, placa do carro, etc.) e deve ser mudada periodicamente. Recomenda-se também que tarefas de rotina não sejam executadas sob a área root. Entrar nessa área apenas para desempenhar tarefas que requerem privilégios especiais e em seguida voltar a conta comum.

2. DISCOS E PARTIÇÕES

2.1. *TERMINOLOGIA*

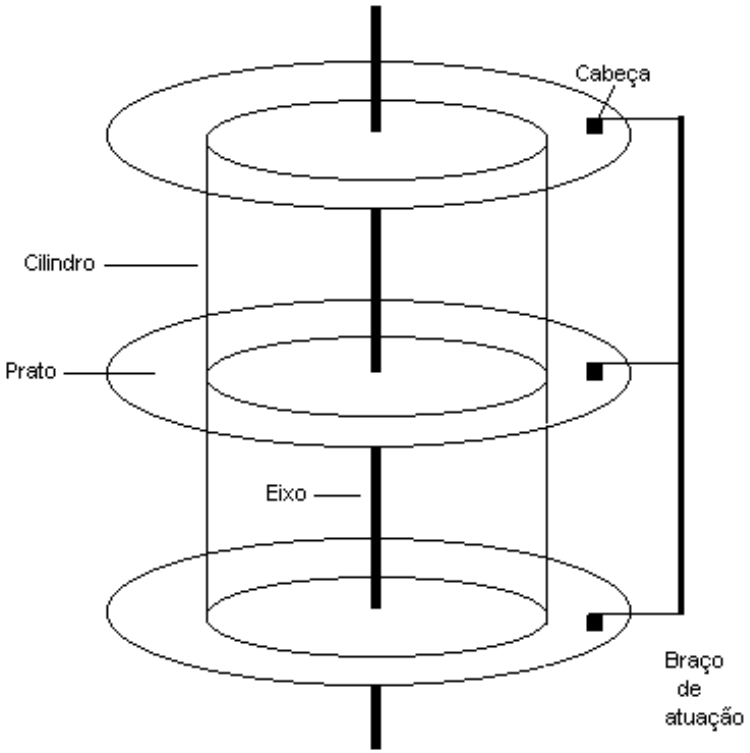
- **Controladoras de discos**
  - **dispositivo que se comunica com a unidadede de disco**
  - **uma controladora suporta várias unidades de discos**
  - **uma estação de trabalho pode ter vários discos**
  - **SCSI - *Small Computer System Interface***
  - **IDE - *Integrated Drive Eletronics***
- **Driver de Dispositivo**
  - **Software residente no kernel que faz a controladora operar**
- **Nome de entrada do dispositivo**
  - **É o nome lógico definido no sistema de arquivo que indica o driver de dispositivo no kernel.**

NOTAS:

Como o nome indica, a controladora de discos é o dispositivo que controla a unidade de disco. A controladora se encarrega de detalhes tais como operações de baixo nível no disco, checagem de erros, movimentar os cabeçotes de leitura e gravação, transferência de dados e disposição dos dados no disco. A controladora de disco provê a interface da unidade de disco com o resto do sistema.

O *driver* de dispositivo (*device driver*) é o *software* que opera a controladora. Este *software* reside no *kernel*. Um dispositivo cujo *software* não houver sido incorporado ao *kernel* não poderá ser usado.

2.2. *ESTRUTURA DO DISCO*

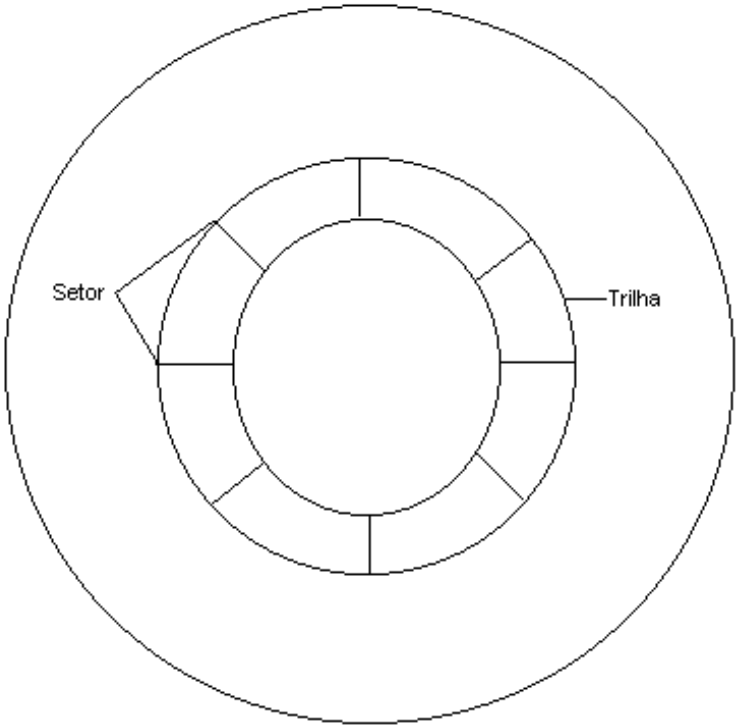


NOTAS:

Uma unidade de disco rígido é fisicamente composto por uma série de pratos ligados a um eixo. A informação contida nestes pratos é processada pelas cabeçotes de leitura e gravação. Estes cabeçotes são acoplados ao braço atuador. No momento da leitura ou gravação de dados, os cabeçotes são movidos conjuntamente pelo braço atuador. O movimento radial é chamado “seeking”, que é uma das razões pelas quais as especificações de disco freqüentemente mencionam o “seek time”. Outros fatores permanecem constantes, quanto mais rápido o “seek time”, mais rápido é o acesso aos dados no disco.

A compreensão da estrutura do disco pode ser útil na sua formatação, pois podem ser necessárias informações como o número de cilindros, cabeças, setores por trilha, velocidade de rotação.

2.3. *PRATO DO DISCO*



NOTAS:

O prato do disco é dividido em trilhas, cilindros e setores. A trilha é a porção do disco que passa debaixo de uma cabeça simples estacionada durante a rotação do disco, formando um anel de 1 bit de largura. O cilindro é composto de um conjunto de trilhas descrita por todas as cabeças em uma simples posição de procura. Cada cilindro é eqüidistante do centro do disco. A trilha é dividida em segmentos de setores, os quais são a unidade básica de armazenamento.

PARTIÇÕES

- **Blocos de disco (setores de 512 bytes)**
- **Cilindros, trilhas, setores**
- **Chamados de dispositivos "raw".**

- **Partições no Solaris:**

#format
partition > print
Current partion table (original sd3):

Part	Tag	Flag*	Cylinders	Size	Blocks
0	root	wm	0 - 51	18.28MB	(52/0/0)
1	swap	wu	52 - 143	32.34MB	(92/0/0)
2	backup	wm	0 - 1150	404.65M	(151/0/0)
3	unassigned	wm	0	0	
4	unassigned	wm	0	0	
5	-	wm	144 - 627	170.16MB	(484/0/0)
6	usr	wm	648-1026	140.27MB	(399/0/0)
7	home	wm	1027-1150	43.59MB	(124/0/0)

*** Flag indica writeable (w), e mountable (m) e umountable(u).**

NOTAS:

Para alguns procedimentos como *backups* e *restores* o uso de partições é mais conveniente.A partição é o dispositivo de disco lógico.

SunOs 5.X (Solaris 2.X) dividiu os nomes dos *devices* em três espaços: físico, lógico, SunOS compatível. Mas para a administração normalmente são utilizados os nomes lógicos.

Os nomes lógicos dos discos contém o número da controladora, o número "target" (alvo) se o disco está no "bus" (portadora), o número do disco e o número do "slice" (partição).O *slice* root é 0, *slice* 1 é swap, *slice* 2 cobre o disco inteiro. Cada *device* de disco tem uma entrada nos diretórios /dev/**dsk** e /dev/**rdsk** para os *devices* de disco bloco e "raw" respectivamente.

Assim, o nome lógico /dev/sd0a do SunOS 4.1.X será no Solaris 2.X:

/dev/dsk/c0t0d0s0

onde:

c0 número da controladora

t0 número de alvo (*target*)

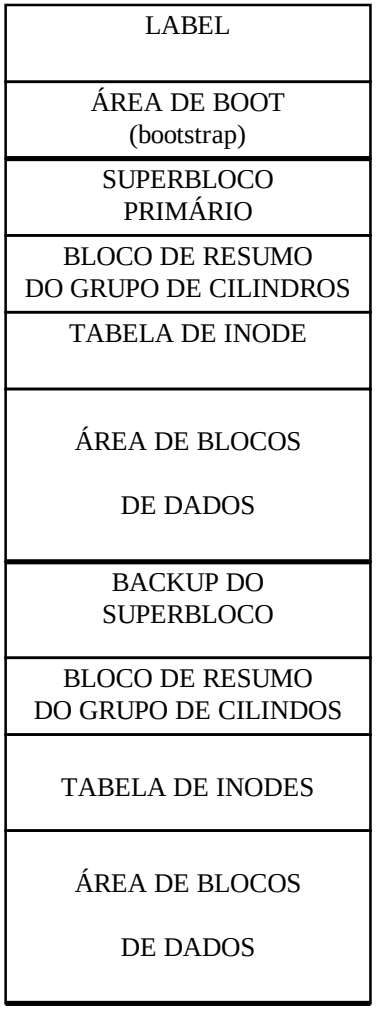
d0 número do disco

s0 número do *slice* (partição)

Este nome é na verdade um *link* simbólico para:

/devices/sbus@1,f8000000/esp@0,8000000/sd@0,0:a (nome físico do *device*).

2.4. *LAYOUT DO DISCO*



NOTAS:

O *label* fica no primeiro setor da primeira partição. Os próximos 15 setores contêm a área de boot.

Seguindo o *label* na partição root e em todas as outras partições vêm as séries de grupos de cilindros. Cada grupo de cilindros contêm um bloco de resumo do grupo de cilindros, uma tabela de *inodes*, e a respectiva área de blocos de dados.

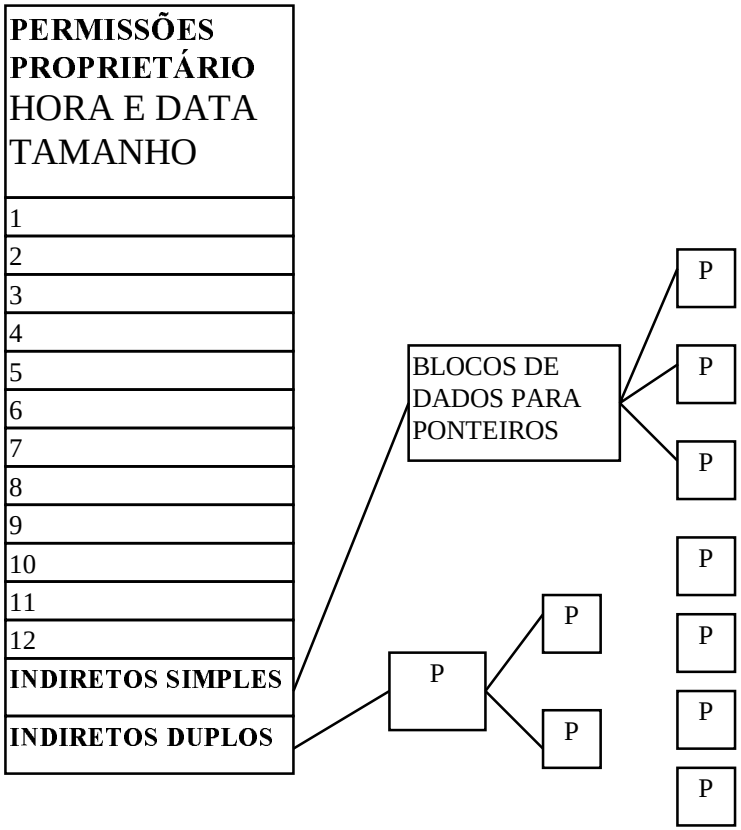
O superbloco primário é mantido na memória e cada grupo de cilindros guarda uma cópia. O tamanho *default* para os blocos de dados é 8192, dividido em 8 fragmentos de 1024 cada.

O bloco de resumo do grupo de cilindros guarda: o número de *inodes* e blocos de dados; ponteiros para o último bloco, fragmento e *inode* usado; o número de fragmentos livres; o mapa de *inodes* usados, o mapa de *inodes* livres.

Um disco pode conter mais de um sistema de arquivos que o UNIX interpreta como um hierarquia de diretórios e arquivos. O UNIX interpreta as solicitações para criar, ler, gravar e apagar arquivos e as executa acrescentando e apagando entradas na lista de *inodes* e blocos livres.

Os diretórios contêm apenas duas informações sobre cada arquivo: o nome e o número do *inode*.

2.5. *INODES*



NOTAS:

Os *Inodes* (*index nodes*) contêm informações a respeito de arquivos e diretórios no sistema de arquivos. A única coisa que o *inode* não contém é o nome do arquivo. O nome do arquivo é mantido no diretório que, também é um tipo de arquivo.

O *inode* contém informações sobre as permissões do arquivo, contagem de ligações (*links*), *timestamp*, blocos duplicados, blocos ruins, associações de tamanho, e ponteiros para os blocos de dados.

O *inode* mantém uma contagem de *links*, que é o número total de referências ao *inode*. Blocos duplicados são blocos apontados por dois *inodes*. Blocos ruins ocorrem quando o número de um bloco está além do limite aceitável. O *inode* mantêm um registro do número de caracteres em um arquivo e o número de blocos a ele associados.

O *inode* possui também apontadores para os blocos de dados dos arquivos. Cada *inode* possui 16 destes apontadores. Os apontadores de 0 a 11 apontam para os 12 primeiros blocos de dados do arquivo (blocos diretos). O apontador 12 aponta para um bloco indireto que aponta para mais 1024 blocos de dados. O apontador de número 13 é um bloco indireto duplo que aponta para 1024 blocos indiretos que por sua vez apontam para 1024 blocos de dados.

3. Startup e Shutdown

3.1. *BOOTING*

- **Auto teste, memória**
- **Identificação**
  - **Modelo**
  - **Tipo de teclado**
  - **Hostid**
  - **Endereço ethernet**
- **Pesquisa o barramento à procura do dispositivo de boot**
  - **SCSI (sd0)**
  - **Network (le0)**
- **PROM lê a trilha de boot (boot block) - SPARC**
- **Carrega o programa de boot (/boot)**
- **O programa de boot lê o kernel**

NOTAS:

Quando a máquina é ligada segue uma seqüência determinada inicialmente pela PROM da placa de CPU. Se todos os testes iniciais forem completados com sucesso, o controle é passado então ao dispositivo de boot. No Sun0s o dispositivo de boot é por *default* a partição a do *drive* 0, mas isto pode ser reprogramado através da EEPROM. Se o *drive* definido não existe ou se o programa de boot (/boot) não for encontrado, o sistema tentará fazer a carga através da rede.

O *startup* a partir do dispositivo de boot procede em estágios até que o Kernel esteja carregado na memória. Neste ponto o controle é passado para o *kernel*.

O *kernel* exibe informações a respeito de seu tamanho e histórico. Ele então pesquisa o barramento para confirmar se os dispositivos especificados no *kernel* realmente existem. Se o dispositivo não existe, então o *kernel* o ignora. Cada dispositivo que precisa ser testado e que não existe atrasa a carga do sistema. Consequentemente, uma das razões pelas quais se recomenda a configuração do *kernel* é diminuir o tempo de boot.

Os dispositivos root, swap e dump são identificados.

- **O Kernel inicializa o sistema**
- **Procura dos dispositivos conhecidos pelo Kernel**
- **Inicializa o processo init**
- **Init lê o arquivo inittab**
- **Executa os scripts RC**
- **Sistema em modo multiusuário**

NOTAS:

O *kernel* invoca o **init** no último estágio do boot. **Init** dá um *fork* para o *background* para rodar durante a existência do sistema. Depois que o **init** é carregado, ele invoca os *shell scripts* RC na seqüência definida por cada sistema. Estes *scripts* executam várias tarefas necessárias para colocar o sistema em modo multiusuário.

Entre estas tarefas está a verificação da consistência dos sistemas de arquivos. Se problemas sérios são detectados pelo **fsck**, o processo de boot pode ser impedido de continuar até que seja feita uma verificação manual dos sistemas de arquivos. Neste caso o sistema é carregado apenas em modo de leitura e o comando **fsck** pode ser utilizado. Entretanto, normalmente as inconsistências são corrigidas automaticamente pelo **fsck**.

Os *scripts* providenciam a montagem dos discos e os *daemons* padrão, depois carregam *daemons* de rede e sistemas de arquivos remotos. Por fim inicializa o sistema em modo multiusuário.

3.2. *INTERRUPÇÕES NO SISTEMA*

SPARC:

- **stop a**

Aborta temporariamente o sistema nos monitores de máquinas SUN

>c
Retoma as atividades do sistema

>sync

Sincroniza os discos e recarrega o sistema (reset)

NOTAS:

A tecla BREAK é usada em sistemas que usam terminais ASCII como console. Esta interrupção joga o console para o nível PROM que aceita vários comandos. O comando “c” (continue) permite a recuperação do sistema paralisado.

Este tipo de interrupção não avisa aos usuários que o sistema será encerrado. Além disto, os sistemas de arquivos não são sincronizados, o que poderá acarretar inconsistências que poderão necessitar correção manual com o programa **fsck**.

O comando **sync** tenta sincronizar os discos e gera um core. Este procedimento deve ser adotado ao invés de simplesmente desligar o sistema após um interrupção com BREAK. O comando **sync** tenta rebootar automaticamente após sincronizar os sistemas de arquivos se for executado no prompt da PROM.

3.3. *PARADA DO SISTEMA*

- **Halt**

Executa imediatamente
Retorna ao prompt da PROM (SPARC)
Sincroniza os discos

- **Reboot**

Executa imediatamente
Sincroniza os discos
Recarrega o sistema
Verifica as partições com fsck

NOTAS:

Os comandos **halt e reboot** sincronizam os discos antes de parar o sistema.

- **Shutdown**

Sincroniza os discos
Alerta os usuários e clientes NFS
Impede logins

- **shutdown [-y] [–g** *segundos***] [-i** *nível_do_sistema***]**

NOTAS:

O comando **shutdown** é um procedimento ordenado de encerramento que alerta os usuários e clientes NFS. Este comando aceita opções, tempo de encerramento e mensagens a serem enviadas aos usuários.

As opções determinam o modo como o sistema será encerrado. A hora de encerramento e uma instrução para rebootar podem ser incluídas na linha de comando.

No caso do Solaris 2.X há diferentes versões do comando **shutdown**.

Exemplo:

shutdown -y -g300 -i0
(leva o sistema para o nível 0 (derruba) em 5 minutos sem perguntas)

3.4. *OPÇÕES DE BOOT*

SPARC:

- **Forma Geral**

> b device(*controller#*, *unit #*, *file#*) *pathname args*

- **Boot default**
  > b

- **Boot em modo single user**
  > b -s

- **Boot através da rede**
  > b le()

- **Boot a partir de uma fita**
  > b st()

- **Boot com opção “ask”**
  > b sd() -a

NOTAS:

O boot pode ser efetuado com várias opções. Estas opções incluem a controladora a partir de onde fazer o boot e o arquivo a ser usado. As máquinas Sun podem fazer o boot a partir de:

Qualquer partição lógica do disco
Um sistema NFS cujo nome seja fornecido pelo *daemon* /bootparamd
O primeiro arquivo de um dispositivo de fita local

O monitor PROM informa quais os dispositivos de boot se você entrar com o comando b ?. Esta lista fornece a ordem pela qual os dispositivos são testados. Não há nenhuma garantia que um dispositivo listado esteja presente ou funcione.

Cada Unix possui um procedimento próprio de *startup*, e suas peculiaridades devem ser procuradas nos manuais específicos.

No PC pode-se bootar em single user usando o *init 1*.

4. INSTALAÇÃO

DE

SISTEMA OPERACIONAL

4.1. *INSTALAÇÃO DE Solaris 2.x*

- **Instalação em SPARC:**
  - **Fazer o boot a partir de um cdrom**
  - **Usar o sysidtool (identificação)**
  - **Providenciar o SunInstall (instalação)**
  - **Escolha entre instalação rápida ou customizada**
  - **Selecione o software desejado**
  - **Para um servidor, configurar os clientes**
  - **Configure os discos de acordo com o software**
  - **Começar a instalação**

NOTAS:

- **Instalação em PC compatível**

  - **Fazer o boot a partir de um disquete de boot do Solaris**
  - **Escolher a opção de instalação pelo Cdrom**
  - **Escolha entre instalação interativa ou customizada**
  - **Selecione os tipos dos dispositivos a serem instalados**
  - **Configure o nome e o IP da máquina**
  - **Configure o name service (other)**
  - **Configure a máscara de rede**
  - **Configure a data e hora**
  - **Selecione a opção de start inicial**
  - **Selecione a categoria da máquina (standalone,server etc)**
  - **Selecione o tipo de instalação (developer, end user etc)**
  - **Configure os discos**
  - **Começar a instalação**

NOTAS:

Caso na instalação a placa de rede não seja reconhecida, verifique se a interrupção está ok, bootando a máquina com o disquete da placa.

Se a sua placa for a NE2000 ou NE2300plus, faça o boot via disquete DOS. Tire o disquete de boot DOS e coloque o disquete de boot do Solaris 2.5.1 e então rode o executável nov2000.bat. Esse executável irá gerar um novo disquete de boot, portanto quando o software perguntar se pode criar esse novo boot, tire o disquete de boot do Solaris 2.5.1 e coloque um disquete apenas formatado na unidade a:.

Fazer o boot novamente, agora com o disquete gerado pelo nov2000.bat e então escolher a opção de instalação via Cdrom.

- **SPARC/PC:**

  - **Para definir rota, editar o arquivo /etc/resolv.conf:**

    nameserver 143.106.xx.xx
    domain ccuec.unicamp.br
    search unicamp.br ccuec.unicamp.br

  - **Para definir a sequência de resolução de nomes, editar o arquivo /etc/nsswitch.conf:**

    hots: dns files [NOTFOUND=return]

  - **Para definir o gateway, editar o arquivo /etc/defaultrouter:**

    143.106.xx.xx

  - **Rebootar a máquina para que ela reconheça as alterações**

NOTAS:

Na definição de rota, é especificado o servidor, o domínio da máquina que está sendo instalada e o search. A seguir um exemplo da aplicação do que foi especificado no search:

telnet obelix

Foi especificado o nome da máquina mas não foi especificado o domínio da mesma. Será verificada então a existência de uma máquina obelix.unicamp.br e caso esta não exista, será verificada a existência de uma máquina obelix.ccuec.unicamp.br.

No nsswitch.conf é definida a ordem de resolução de nomes, para ver se a máquina desejada existe na rede (exemplo se obelix.unicamp.br existe). Com a definição que foi exibida ao lado a ordem é a seguinte:

- Verifica na máquina DNS
- Se não encontrou verifica nos arquivos configurados na máquina local
- Se não encontrou retorna com mensagem de máquina não localizada.

5. CONFIGURAÇÃO DO KERNEL

5.1. *PORQUE CONFIGURAR O KERNEL*

- **Obter maior velocidade**
- **Reduzir uso de memória**
- **Adequar-se à configuração do sistema**
- **Acrescentar novos dispositivos de hardware**
- **Acrescentar novos softwares**

NOTAS:

O *kernel* que vem junto com o sistema é configurado para suportar todos os dispositivos de uma determinada arquitetura. na realidade, nenhuma máquina terá conectada a ela todos estes dispositivos a um só tempo. O código do *kernel* ocupa espaço na memória. para reduzir o tamanho do *kernel*, elimine os módulos que não são necessários, reduzindo assim memória para uso de programas e Consequentemente, aumentando a performance do sistema.

Durante o boot, o sistema pesquisa cada dispositivo do *kernel* para confirmar sua existência. Se o dispositivo não responder, o sistema desiste após algum tempo e passa para o próximo dispositivo. para eliminar este período de espera requerido pelo *timeout*, configure o *kernel* de forma a que apenas os dispositivos existentes no sistema sejam testados.

Hardware adicional - acréscimo de discos, placas, mudanças para monitores coloridos - requerem suporte do *kernel*. À medida que o sistema cresce , o *kernel* deve ser reconfigurado para suportar o novo *hardware* que lhe é acrescentado. Isto também pode se aplicar ao *software*. Programas aplicativos podem incluir *drivers* de dispositivos que precisam ser incluídos no *kernel*. Estas informações são normalmente encontradas na documentação dos distribuidores independentes de *software*.

5.2. *Configuração do KERNEL*

SunOS 5.x.x

- **Configurar o /etc/system**

NOTAS:

O kernel é alterado automaticamente durante o próximo boot.

6. AMBIENTE OPERACIONAL

6.1. *Ambiente Operacional*

- **Conjunto de variáveis que definem ou controlam determinados aspectos da operação**
- **Variáveis lidas do arquivo /etc/profile ou /etc/default/login ou ainda definidas por default**

NOTAS:

O ambiente operacional é primeiramente o conjunto de variáveis que definem ou controlam certas características da operação do sistema. A maior parte destas variáveis é definida durante a inicialização do sistema, sendo que suas definições são lidas do arquivo /etc/profile ou definidas por default.

O shell utiliza dois tipos de arquivos de inicialização quando um usuário loga no sistema. Ela avalia os comandos contidos nestes arquivos e então os executa para definir o ambiente operacional. Os arquivos possuem funções similares, com a exceção de que o arquivo /etc/profile controla variáveis aplicáveis a todos os usuários ao passo que o arquivo .profile permite a customização individual do ambiente.

6.2. *Profiles*

- **Arquivos utilizados pela shell para definição do ambiente operacional:**
  - **/etc/profile**
  - **/etc/default/login**
  - **$HOME/.profile (quando o shell a ser usado for Korn shell....)**
  - **$HOME/.login (quando o shell a ser usado for C shell)**

NOTAS:

Os primeiros arquivos que o sistema lê durante o login de um usuário são os arquivos /etc/profile ou /etc/default/login. Estes arquivos controlam variáveis tais como:

- Variáveis exportáveis
- Máscara de criação de arquivos
- Tipos de terminal
- Mensagens utilizadas para informar o usuário da chegada de novas mensagens eletrônicas

O administrador do sistema configura o arquivo /etc/profile para todos os usuários do sistema. Este arquivo tem acesso permitido apenas para o usuário *root*.

O segundo arquivo que o sistema operacional usa durante o login é o arquivo .profile(quando é usado Korn shell) ou .login(quando é usado C shell) encontrados no diretório *home* do usuário. Estes arquivos permitem que cada usuário customize o seu ambiente de acordo com suas preferências pessoais. Os comandos contidos neste arquivo /etc/profile. Entre outras coisas os arquivos .profile e .login são usados para:

- Definir o shell a ser usado
- Aparência do prompt
- Variáveis de ambiente ($PATH por exemplo)

6.3. *Shells*

- **Programa**
- **Aceita e executa comandos**
- **Linguagem de programação**
- **Poderosa e flexível**
- **Disponíveis com o Solaris 2.5.1 (entre outras)**
    - **C-shell**
    - **Korn Shell**

NOTAS:

O shell é a interface entre o sistema e o usuário. É um programa que aceita e executa comandos.

Comandos de shell normalmente executam uma determinada tarefa muito bem. A capacidade de combinar estes comandos para situações específicas dá ao programador um grande controle sobre o sistema.

O sistema Solaris 2.5.1 vem com a Bourne shell, C-shell, Korn shell, shell restrita e trusted shell. O usuário pode escolher o shell a que melhor se adapte.

6.4. *C Shell*

- **/bin/csh**
- **% (prompt)**
- **Desenvolvida em Berkley**
- **Sintaxe semelhante à linguagem C**
- **Possui muitos comandos e variáveis úteis**
- **Aliases**
- **Controle de tarefas**

NOTAS:

Como o sistema Unix foi escrito em linguagem C e como era freqüentemente utilizado por programadores C, era de se esperar que se criasse um shell que aceitasse comandos com sintaxe semelhante à linguagem C.

O C shell possui funções que não estão disponíveis na Bourne shell, como por exemplo um histórico dos comandos emitidos com resubmissão e uma função para criação de comandos customizados.

Os aliases são uma maneira de se criar comandos customizados.

O controle de tarefas fornece a facilidade de se suspender e de se retomar tarefas, colocar tarefas em *background* e *foreground.*

Você pode encontrar o arquivo .cshrc no diretório home do usuário. Ele contém o controle de recursos da C shell

6.5. *Korn Shell*

- **/bin/ksh**
- **$ (prompt)**
- **Possui a sintaxe e as funções da Bourne shell**
- **Funções da linguagem C**
- **Aliases**
- **Controle de tarefas**
- **Histórico de comandos**
- **Edição da linha de comandos**
- **Arquivo de ambiente**

NOTAS:

O Korn shell combina a sintaxe da Bourn shell com as funções adicionais, tal como aliases e histórico de comandos, à semelhança do C shell.

O histórico dos comandos executados é mantido em um arquivo designado pela variável HISTFILE (default $HOME/.sh_history). O tamanho do arquivo é determinado pela variável HISTSIZE (default é 128 comandos). A variável FCEDIT é usada para definir o editor utilizado para editar comandos.

A edição da linha de comandos pode ser feita usando ou o editor vi ou emacs. “set –o vi” ou “set –o emacs”especificam qual editor será usado.

Quando as variáveis são definidas, elas estão disponíveis apenas para o processo no qual elas foram definidas a menos que elas sejam exportadas. Da mesma forma, aliases e funções somente estão disponíveis no processo no qual elas foram criadas a menos que algo especial seja feito.

7. PROCESSOS

7.1. *Visão Geral*

- **Programas ou comandos sendo executados**
- **Hierarquia pai-filho**
- **Pid – multiprocessamento – scheduler**
- **Foreground – background**
- **Daemons**
- **zombie**

NOTAS:

O que é um processo? É um programa ou comando que está sendo executado pelo computador, o qual pode executar mais de um processo ao mesmo tempo(multiprocessamento).

A criação de processos obedece à hierarquia pai-filho.Assim todo processo pai pode ter mais que um processo filho, e todo processo filho possui apenas um pai. O processo pai é aquele que foi criado a partir de um programa ou comando e o processo filho é aquele criado por um processo do tipo pai.

Por ser um sistema de multiprocessamento, o sistema associa um número a cada processo (***pid***), que é diferente de todos os números já associados a processos durante o tempo relativo da entrada da máquina no ar até o momento de sua queda. Cada processo utiliza uma determinada quantidade de tempo disponível do sistema, de acordo com uma política de escalonamento e também de acordo com a prioridade dos processos.

Processos considerados *foreground*, são aqueles criados através da linha de comando e que precisam de interação com o usuário. Durante a sua execução nenhum outro comando pode ser executado. Processos *background* são aqueles que executam independentemente da criação pela linha de comando, sem a interação com o usuário.

Daemons são processos criados no momento de boot da máquina e que continuam executando até o encerramento do sistema. Estes tipos de processos executam serviços do sistema e estão disponíveis sempre para mais de um usuário ou tarefa. Eles somente podem ser criados ou parados pelo usuário root.

Zombies são processos “stopped” mas que ainda são reconhecidos pela tabela de processos. Este tipo de processo não utiliza nenhum recurso do sistema operacional. Eles são liberados somente quando o processo pai também é “stopped”, ou o sistema é reinicializado.

7.2. *Comandos úteis*

- **Linha de comando &**

  Exemplo:
  %find . –name telnet –print &
  %find . –name telnet –print

- **Para ver jobs background**
  % jobs –l

  A linha de resposta será a seguinte:
  [1] + 250 Running find . –name telnet -print

- **Para fazer com que jobs background tornem-se foreground**
  %fg *pid*
  No exemplo acima o *pid* é 1

- **Para parar um processo background**
  %stop *jobid*
  No exemplo acima o *jobid* é 250

- **Para passar um processo de foreground para background**
  Teclar <CTRL> Z e então digitar bg <ENTER>

NOTAS:

8. GERENCIAMENTO DE USUÁRIOS

8.1. *Criação de usuários*

- **Os três componentes de uma conta de usuário**
  - **/etc/passwd e /etc/shadow**
  - **diretório *home***
  - **arquivos de configuração**
    - **.cshrc ou .kshrc**
    - **.login ou .profile**
- **Grupos de usuário (/etc/group)**

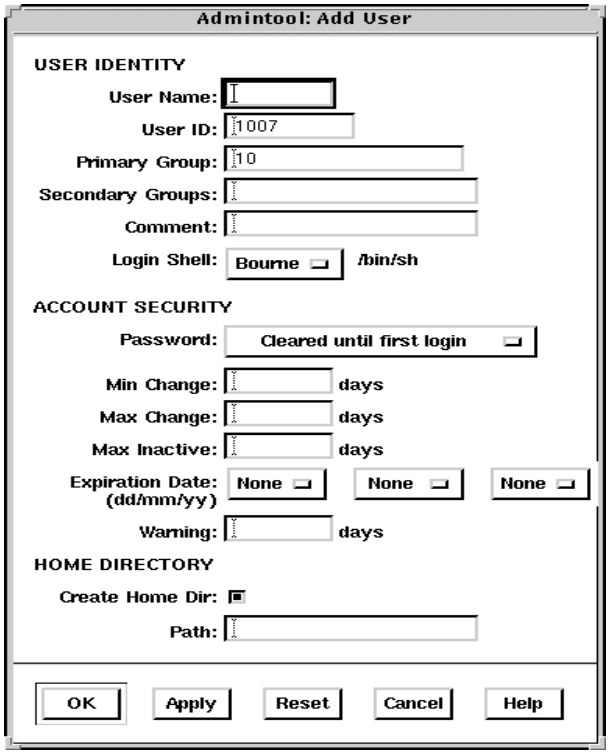
NOTAS:

O diretório home do usuário deve ser criado antes da criação do próprio usuário. O owner deste diretório também deve ser alterado para que passe a ser o próprio usuário e não mais o root com o seguinte comando:

%chown –R *usuário usuário*

Caso isso não seja feito o usuário não conseguirá acessar seu próprio home.

- **No Solaris 2.x as contas devem ser abertas preferencialmente através do programa de administração (admintool)**



NOTAS:

Além da abertura de contas o admintool permite modificar vários outros mapas importantes na administração:

- usuários
- grupos
- hosts
- impressoras
- portas serias
- software

Tanto no Solaris 2.x as contas de usuários podem ser abertas de duas outras formas

- Pelo comando vipw
- **Pelo comando useradd**

8.2. *Administração de usuários*

- **Alteração, deleção ou consulta das contas dos usuários também devem ser feitas pelo Admintool.**

- **Definir os atributos defaults para as contas dos usuários**

- **A alteração de senha pode ser feita pelo Admintool ou pelo comando *passwd***

NOTAS:

9. Sistema de Arquivos

9.1. *Sistemas de Arquivos*

- **Estrutura de arquivos e diretórios**
- **Superblocos, blocos de grupo de cilindros**
- **Bloco de boot**
- **Criado pelos comandos mkfs ou newfs**
- **Precisa ser "montado"**
- **Reparado pelo comando fsck**

NOTAS:

O sistema de arquivos é uma estrutura de dados que contém arquivos e diretórios. Os sistemas de arquivos (um por partição são criados pelos comandos ***mkfs*** ou ***newfs***.Problemas no sistema de arquivos podem ser corrigidos pelo comando ***fsck***

9.2. *MANUTENÇÃO DE SISTEMA DE ARQUIVOS*

- **Os sistemas de arquivos podem vir a se danificar devido a:**
  - **Parar o sistema sem sincronizar os discos**
  - **Não verificar e reparar inconsistências nos sistemas de arquivos durante o startup**
  - **Erros no disco**
  - **Falhas de hardware**

NOTAS:

9.3. *FSCK*

- **O programa fsck checa:**
  - **Inconsistências nas informações do superbloco**
    - **Tamanho do sistema de arquivos**
    - **Números de inodes**
    - **Contagem de blocos livres**
    - **Contagem de inodes livres**
  - **Inconsistências no mapa de blocos do grupo de cilindros**
    - **Blocos livres requisitados por arquivos**
    - **Total livres+blocos requisitados=total de blocos**
  - **Inconsistências nos inodes**
    - **Um inode está alocado ou livre, nunca ambos**
    - **Contagem de links correta**
    - **Inodes não referenciados**

NOTAS:

Inconsistências lógicas são quase sempre descobertas pelo programa fsck ao checar a integridade dos sistemas de arquivos, e são normalmente ocasionadas por quedas do sistema operacional. O programa **fsck** pode efetuar os reparos tanto interativamente quanto automaticamente (através da opção -p).

O programa **fsck** checa os sistemas de arquivos identificados no arquivo /etc/fstab como do tipo 4.2. O arquivo /etc/fstab pode ser usado para descobrir a ordem em que os sistemas de arquivos são examinados pelo programa fsck.

O programa **fsck** se baseia no fato de que não devem haver inconsistências nos sistemas de arquivos. Por exemplo, o número de inodes é especificado no superbloco. Qualquer informação que contrarie este número é considerado suspeita. O programa fsck tenta então consertar o sistema de arquivos baseado no tipo de problema encontrado.

Arquivos ou diretórios órfãos são reconectados aos sistemas de arquivos sob o diretório lost+found. Cada sistema de arquivos possui um diretório com esse nome.

O programa fsck verifica as seguintes inconsistências:

Blocos requisitados por mais de um inode, ou requisitado por um inode na lista de inodes livres.

Blocos requisitados por um inode ou lista de blocos livres fora dos limites do sistema de arquivos.

Contagem de links incorreta.

Tamanho de diretórios incorretos.

Informação sobre inodes incorreta.

Blocos não identificados em lugar nenhum

Arquivo apontando para um inode livre, ou um número de inode fora do limite.

Verificação dos superblocos: mais blocos por inodes do que existem no sistema de arquivos.

Formato da lista de blocos livres incorreto.

Total de blocos livres ou inodes livres incorretos.

- **O programa fsck verifica, mas não faz correções em um sistema de arquivos ativo**

- **Para verificar a integridade de todos os sistemas de arquivos listados no arquivo /etc/fstab :**
  #fsck

- **Exemplos:**

Checar o sistema de arquivos /dev/dsk/c0t0d0s0
#fsck dev/dsk/c0t0d0s0

Checar o sistema de arquivos na partição dev/dsk/c0t0d0s0 usando o superbloco alternado 32 (SPARC)
#fsck -b 32 dev/dsk/c0t0d0s0

NOTAS:

O programa fsck invocado sem nenhum argumento checa todas as partições contidas no arquivo fstab. O programa fsck aceita *mount points* como argumento e determina através daí o dispositivo correto a checar. A verificação do dispositivo "raw" é sempre mais rápida.

Se for utilizada opção -y, todas as correções são feitas sem confirmação. Esta opção não é aconselhável, se todos os dados da partição forem importantes. O programa fsck faz as modificações diretamente no disco.

O programa fsck pode ser instruído a utilizar um superbloco alternado. Para determinar os superblocos existentes use o comando newfs com a opção -N <número>. (Muito cuidado, pois o comando newfs sem opção -N apagará todos os dados).

- **Fases do FSCK:**

**** Phase 1 - Check Blocs and Sizes**
3788627 BAD I=66
**** Phase 2 - Check pathnames**
DUP/BAD I=66 OWNER=root MODE 100755
SIZE=76409 MTIME=Mar 3 15:29 1995
FILE=/usr/bin/kawan
REMOVER?y
**** Phase 3 - Check connectivity**
**** Phase 4 - Check Reference Counts**
BAD/DUP I=66 OWNER=root MODE=100755
SIZE=76409 MTIME=Mar 3 15:29 1995
FILE=/usr/bin/kawan
CLEAR?y
UNREF FILE I=531 OWNER=root MODE=100600
SIZE=0 MTIME=Feb 28 11:35 1995
RECONECTY?y
**** Phase 5 - Check Free List**
137 BLK(S) MISSING
BAD FREE LIST
SALVAGE?y
**** Phase 6 - Salvage Free List**
1398 files blocks 6546 free

NOTAS:

Fase 1: Checa Blocos e Tamanhos (checa inodes procurando inconsistências)
Fase 2: Checa *Path-Names* (checa diretórios e inconsistências de inodes)
Fase 3: Checa Conectividade (checa se todos diretórios estão conectados no sistema de arquivo)
Fase 4: checa Reference Counts (compara informações de contagem de link da fase 2 e 3, corrigindo discrepância)
Phase 5: Checa grupos de cilindros (checa blocos livres e o mapa de inodes usados procurando por inconsistências)

No exemplo ao lado, o programa fsck descobriu um inode duplicado para um arquivo chamado /usr/bin/kawan. Neste caso, o superusuário decidiu desconectar o inode e instruiu o programa fsck a prosseguir e limpa-lo. Ao final, o programa descobriu que existem blocos faltando na lista de blocos livres. Ele pergunta então se a lista de blocos livres deve ser salva. Com a resposta afirmativa o programa fsck prossegue e salva a lista.

Normalmente o programa fsck é executado em cinco fases, mas neste caso existe uma fase adicional.

Algumas vezes, após o fim da execução do programa fsck aparece uma mensagem do tipo:

**** BOOT UNIX (NO SYNC) ****

Isto acontece quando o sistema de arquivos montado foi modificado. Se o sistema não é reinicializado neste ponto, a modificações feitas pelo programa fsck serão perdidas se os dados no superbloco forem gravados novamente no disco.

Interrompa (L1-A, BREAK) e reinicialize o sistema imediatamente. Não espere que o sistema se atualize sozinho.

9.4. *MONITORANDO ESPAÇO EM DISCO*

#df

O comando df (disk free) mostra o tamanho total (não incluindo os 10% adicionais), número de kbytes usados e disponíveis, porcentagem total usada, e *mount point* de cada sistema de arquivos.

#du

O comando du (disk usage) mostra o número de blocos contidos em cada subdiretório do diretório corrente.

#du -a | sort -nr

Lista todos os arquivos neste diretório em ordem de tamanho do maior para o menor incluindo o caminho completo

NOTAS:

9.5. *FIND*

#find / -name '*core' -print

Exibe todos arquivos cujos nomes terminem em core a partir

#find / -mtime +90 -print

Exibe todos os arquivos não modificados por noventa dias.

#find / -name '#*' -mtime +7 -print

Exibe todos os arquivos iniciados por "#" cuja data de última modificação exceda 7 dias

NOTAS:

9.6. *CRONTAB*

- **Deamon : /usr/etc/cron**

- **A ser executado: /var/spool/cron/crontab**

0,30 * * * * /bin/date > /dev/console

0 0 * * * calender -

15 0 * * * /usr/etc/sa -s > /dev/null

0,30 * * * * /etc/dmesg ->> /usr/adm/messages

NOTAS:

O daemon /usr/etc/cron é usado para executar comandos em horários pré determinados. Ele lê arquivos contidos no diretório /var/spool/cron/crontabs. Estes arquivos pertencem a usuários do sistema; os arquivos determinam quais programas ou scripts serão executados e quando.

Os registros nestes arquivos contêm cinco campos. São eles:

Campo	Valores permitidos
minuto	0-59
hora	0-23
dia do mês	1-31
mês	1-12 (Janeiro – Dezembro)
dia da semana	0-6 (Segunda – Domingo)

Comando	
n	O comando é executado quando o valor do campo é n
n,p,q	O comando é executado quando o valor do campo é n, p, q.
n-q	O comando é executado quando o valor do campo é n-p
*	O comando é executado para todos os valores possíveis do campo

/var/spool/cron/cron.allow
Se esse arquivo existir, apenas os usuários nele contidos são autorizados

/var/spool/cron/cron.deny
Se o arquivo cron.allow não existir, os usuários contidos no arquivo cron.deny não são autorizados a executar o comando crontab

#crontab -e
Edita o arquivo crontab do usuário

#crontab -l
Lista o arquivo crontab do usuário

crontab *nome_do_arquivo*
Edita o arquivo crontab especificado

NOTAS:

9.7. *AT*

```
%at –c 23:30

at>echo “oi, eu funcionei” > arq

at>^D

%atq

Rank   Execution Date        Owner   Jobs#  JobName
1st    Dec 22, 1992 01:00    aldo    2350   Stdin

%atrm 2350
  2350 removed
```

NOTAS:

9.8. *USO DE QUOTAS DE DISCO*

- **Adicionar opção de quota para o sistema de arquivo**

../c0t0d0s7 /home ufs 1 yes rq

- **Criar o arquivo *quotas* no diretório pai da partição que irá trabalhar com quotas**
  #touch /home/quotas

- **Configurar quotas para cada usuário**
  #edquota marcelo
  fs /home blocks (soft=2500,hard=4500)
  inodes (soft=80, hard=1250)

- **Ligar a verificação de quota**
  #quotaon -av

- **Verificação da quota de disco de um usuário**
  #quota -v marcelo

NOTAS:

10. BACKUP E RECUPERAÇÃO

10.1. *BACKUPS*

- **Uma das tarefas mais importantes do administrador de sistemas**
- **dump / restore (ufsdump / ufsrestore)**
- **tar**
- **cpio**
- **dd**

NOTAS:

Uma das tarefas mais importantes do administrador de sistemas, é garantir que os dados a ele confiados nunca serão perdidos. Acidentes freqüentemente acontecem e as conseqüências podem variar de um simples arquivos perdido a anos de trabalho jogado fora.

Existem vários utilitários no Unix que permitem que se façam cópias do sistema. A escolha de qualquer um deles vai depender do gosto pessoal de cada administrador.

Abordaremos aqui com mais detalhes os comandos **dump** e **restore**, usados respectivamente para armazenar e restaurar arquivos.

O comando **dump** permite que se façam *backups* completos (*full*) de sistemas de arquivos ou de diretórios. Permite também *backup* incrementais. Novamente, a decisão de quando se fazer *backups* completas e incrementais vai depender das particularidades de cada instalação e do administrador de sistemas.

10.2. *TAR*

- **É o comando de cópia de diretórios mais utilizado no UNIX**
- **Cria arquivos de arquivos e possui múltiplos usos**
- **Sintaxe**

tar [cxtvf] *saída/entrada* [*diretório*]

***saída/entrada* pode ser um device ou um arquivo**

-c copia para a saída
-x extrai arquivos da entrada
-t lista a entrada
-v mostra o comando funcionando
-f opção para identificar o dispositivo de i/o

- **Exemplos**

tar cvf /dev/rmt/0 . (copia diretório corrente para fita)

tar tvf backup.tar . (lista o conteúdo de backup.tar)

NOTAS:

10.3. *CPIO*

- **Menos utilizados, também permite criar e manipular sistema de arquivos**

- **Sintaxe**

  cpio -[ioc] [><] *saída/entrada*

  ***saída/entrada* pode ser um device ou um arquivo**

  -o define o nome da saída
  -i define o nome da entrada
  -c cria um cabeçalho portavel entre sistemas

- **Exemplos**

ls | cpio -oc > /dev/rfd0 (copia diretório corrente para disquete)

cat jtr2.cpio | cpio -ic (extrai arquivos de jtr2.cpio)

cpio -ic < /dev/rfd0 (lê backup em disquete no formato cpio)

NOTAS:

10.4. *DD*

- **Trabalha com arquivos, mas é mais usado para trabalhar com backup de partições, disquetes, fitas e conversões**

- **Sintaxe**

  dd *[operador=valor] [operador=valor] ...*

  Principais operadores

  if=file (entrada)
  of=file (saída)
  bs=tamanho (configura o tamanho do bloco)
  conv=opção (faz conversões)

- **Exemplos**

dd if=boot.flp of=/dev/rfd0 (copia imagem para flop)

dd if=/dev/rmt/0 of=/dev/rmt/1 (duplica fita)

dd if=teste.txt of=TESTE.TXT conv=ucase

NOTAS:

10.5. *PREPARATIVOS PARA O BACKUP*

- **Antes de iniciar um backup certifique-se que não existem usuários no sistema**
- **Rode o programa *fsck* para corrigir inconsistências no sistema de arquivos**
- **Se existirem usuários logados use os comandos wall, rwall e shutdown para anunciar o encerramento do sistema**
- **O sistema deve estar em modo monousuário durante backups de nível zero**
- **Não faça backup das áreas de swap**

NOTAS:

O programa **fsck** verifica a integridade do sistema de arquivos e faz correções onde for possível. Este programa deve também ser rodado sempre que se fizer um restore de um sistema de arquivo completo.

O programa **who** informa se existem usuários logados no sistema. Se um usuário estiver logado com um longo tempo de ociosidade, provavelmente esqueceu de dar **logoff**.

Para as paradas programadas aconselha-se avisar o usuário por meio do arquivo /etc/motd (*message of the day*). A mensagem contida neste arquivo é lida durante o processo de *login*. O programa **wall** (*warn all*) serve para divulgar uma mensagem aos usuários logados no sistema. O programa **rwall** permite propagar uma mensagem aos clientes de um servidor.

O programa **shutdown** faz o encerramento do sistema em um tempo predeterminado. Mensagens são enviadas periodicamente aos usuários logados e o intervalo entre as mensagens vai diminuindo à medida que se aproxima o horário de encerramento.

É recomendado que os **backups** de nível zero sejam realizados em modo monousuário de modo a assegurar o mínimo de inconsistência possível entre o **backup** e o sistema de arquivos.

Aconselha-se pela mesma razão, que os backups incrementais sejam sempre realizados quando a taxa de utilização do sistema for mínima.

10.6. *BACKUP DE SISTEMAS DE ARQUIVOS*

- **A unidade de fita será reconhecida no momento de instalação do Solaris 2.x**

- **O comando padrão usado para backup no Solaris 2.x é o UFSDUMP**

- **Sintaxe**

ufsdump [opções] [*saída*] [sistema_de-arquivos]

Principais opções:

0-9 nível de backup
u atualliza arquivo de log (/etc/dumpdates)
f identifica argumento de saída

- **Exemplos**

Ufsdump full da partição /home:

#ufsdump 0uf /dev/rmt/0 /home

Ufsdump em uma fita remota

#ufsdump 0uf obelix:/dev/rmt/0 /home

NOTAS:

O comando ufs**dump** permite a cópia de sistemas de arquivos inteiros. O nível do **dump** varia de 0 a 9. O nível 0 é o mais baixo e copia integralmente o sistema de arquivos e o nível 9 é o mais alto. Um dump incremental salva todos os arquivos modificados desde a data do último dump de nível mais baixo.

Um backup de nível 0 (full) precisa ser feito antes que se façam dumps incrementais.

As opções aceitas pelo comando **dump** são:

0-9 Este número especifica o nível do dump.
f Especifica o dispositivo onde será feita a gravação do dump.
u Atualiza o arqu ivo /etc/dumpdates
c Especifica unidade de cartucho.
b Blocagem a ser utilizada. O valor default é 20 blocos para unidade de fitas de 1/2” 2 126 para unidades de fita de 1/4”. A blocagem default para gravação em densidade de 6250 BPI é de 64 blocos. Quanto maior o fator de blocagem mais rápido será o dump. O tamanho do bloco para fitas é de 512 bytes.
s Especifica o comprimento da fita. O default para fitas de 1/2” é de 2300 pés, 700 pés para fitas cartucho de 150M, 6000 pés para fita de 8mm (2 Gbytes) e 13000 pés para fita de 8mm (5 Gbytes).
D Especifica a densidade de gravação. O default para fitas de 1/2” é de 1600 bpi, para fitas de 1/4” é de 1000, para fitas de 2 e 5 Gbytes é de 54000 bpi.

Para fazer **dumps** de sistemas de arquivos em unidades de fita remotas, certifique-se que o arquivo /.hosts da máquina remota contém uma entrada com o nome de sua máquina.

10.7. *BACKUP DE VÁRIOS SISTEMAS DE ARQUIVOS EM UMA FITA*

- **Utilize a opção norewind**

  #ufsdump 0uf /dev/rmt/0n /usr

- **Faça o dump seguinte:**

  #ufsdump 0uf /dev/rmt/0 /home

NOTAS:

Se for necessário que se faça o dump de mais de um sistema de arquivos utilizando a mesma fita, utilizar a opção **n** (norewind) na definição do dispositivo da fita.

Utilizando o device /dev/null nenhum dump será realizado, mas pode-se conseguir uma estimativa de consumo de fitas.

O comando **df** também pode ser utilizado para a mesma finalidade.

A opção norewind suprime o rewind após o encerramento do dump da partição. Assim, o dump da próxima partição será feito a partir do ponto da fita onde termina o primeiro backup.

Entretanto, se o segundo dump não utilizar a opção norewind, a fita será rebobinada.

É necessário um comando separado para cada partição.

A opção u faz com que o programa dump insira um registro no arquivo /etc/dumpdates especificando a partição que foi copiada, o nível e a hora do dump.

```
cat /etc/dumpdates
/dev/dsk/c0t0d0s0   1  Wed May 22   04:33:01  1997
/dev/dsk/c0t0d0s3   1  Wed May 22   04:33:01  1997
/dev/dsk/c0t0d0s7   1  Fri  Apr 15   18:50:32  1997
```

10.8. *RECUPERAÇÃO DE BACKUP*

- **O comando para recuperação de backups feitos com o ufsdump é o ufsrestore**

- **Sintaxe**

  ufsrestore [opções] [arquivo_de_backup]

  Principais opções: t,i,r,x,v,f

- **Recuperação de backup**

  #ufsrestore xvf /dev/rmt/0 *arquivo(s)*
  Specify next volume#:n
  Set owner/mode for '.' [y/n]y

- **Restore através da rede**

  ufsrestore salvia:/dev/rmt/0 *arquivo(s)*

- **Restore em fita com múltiplos backup**

  ufsrestore xvS /dev/rmt/0 2

NOTAS:

Um usuário pode perder um arquivo, ou sistema de arquivos inteiros devido a falta de cuidado, um programa mal escrito ou com erros, ou, através do uso impróprio de utilitários. Neste caso o administrador de sistemas deve ser capaz de reparar o erro e restaurar os arquivos ou os sistemas de arquivos perdidos. Isto pressupõe a existência de um backup recente em fita magnética. Envolve também descobrir em que fita se encontram os arquivos desejados. A opção do comando restore fornece uma listagem de todos os arquivos contidos em uma fita de dump.

As opções aceitas pelo restore são as seguintes:

t Lista o conteúdo do arquivo de dump.
i Invoca o programa restore em modo interativo.
r Restore recursivo. Todos os arquivos na fita são restaurados.
x restaura apenas os arquivos especificados na linha de comando
v Roda em modo verboso, exibindo o nome de todos os arquivos restaurados.
f Especifica o dispositivo de fita magnética. Obrigatório o uso quando usado o comando restore.

A organização de fitas de backup é muito importante. Nem sempre a versão mais recente de um arquivo é a requerida. Todas as fitas de dump devem estar numeradas e datadas. O restore não deve ser feito no mesmo diretório onde se encontrava o arquivo original. Você pode escolher entre criar um novo diretório ou fazer o restore no arquivo /tmp. Após os arquivos tiverem sido restaurados copie-os para o lugar correto. Certifique-se também que o modo do arquivo esteja correto (ownership/permissions).

Para restaurar um arquivo de uma fita que se encontra montada em um sistema remoto certifique-se que existe em seu sistema, no arquivo /.hosts, uma entrada com o nome do sistema remoto. Isto porque o programa restore é rodado na área root.

- **Restore Interativo**

```
#cd /tmp
#ufsrestore ivf /dev/rmt/0
restore>ls
restore>cd /usr
restore>add arquivo
restore>extract
restore>delete arq
restore>verbose
restore>quit
#
```

NOTAS:

- **RESTORE DE UM SISTEMA DE ARQUIVOS**

Prepare a partição rodando newfs:
#newfs /dev/rdsk/c0t0d0s7

Verifique integridade da partição
#fsck /dev/rdsk/c0t0d0s7

Monte a partição em uma área temporária
#mount /dev/dsk/c0t0d0s7 /home

Restore o sistema de arquivos
#cd /home
#ufsrestore rvf /dev/rmt/0
#rm restoresymtable

Verifique a integridade do sistema de arquivos
#fsck /dev/rdsk/c0t0d0s7
#

NOTAS:

Certas situações por vezes exigem que um sistema de arquivos seja restaurado em sua integridade. O disco pode ter se danificado a tal ponto que a sua troca seja requerido, a instalação de novos programas pode requerer o redimensionamento do disco, etc.

- **RESTORE DO /ROOT**

Carregue o sistema a partir de uma fita ou CDROM
>b sd(0,6,2) (SunOS e Solaris em CD na Sparc 1+)
>b cdrom -sw (Solares 2.x em Sparc > 2)

Se houver problemas com o disco, rode o programa format para analisar e corrigir o problema

Utilize o comando newfs para recriar o sistema de arquivos /root
#newfs /dev/dsk/c0t3d0s0 (Solares 2.x)

Verifique o sistema de arquivos criado
#fsck /dev/dsk/c0t3d0s0

Monte o sistema para fazer o restore
#mkdir /mnt
#mount /dev/ dsk/c0t3d0s0 /mnt
#cd /mnt
#ufsrestore rf /dev/rmt/0

Remova o arquivo restoresymtable após o final do restore
#rm restoresymtable

NOTAS:

Remova o arquivo restoresymtable após o final do restore
#rm restoresymtable

Desmonte o sistema de arquivos
#cd /
#umount /mnt

Instalar o bloco de boot após o ufsrestore do sistema de arquivos root
#cd /usr/lib/fs/ufs
#installboot bootblk /dev/rdsk/c0t3d0s0

Verifique o sistema de arquivos criado
#fsck /dev/rdsk/c0t3d0s0

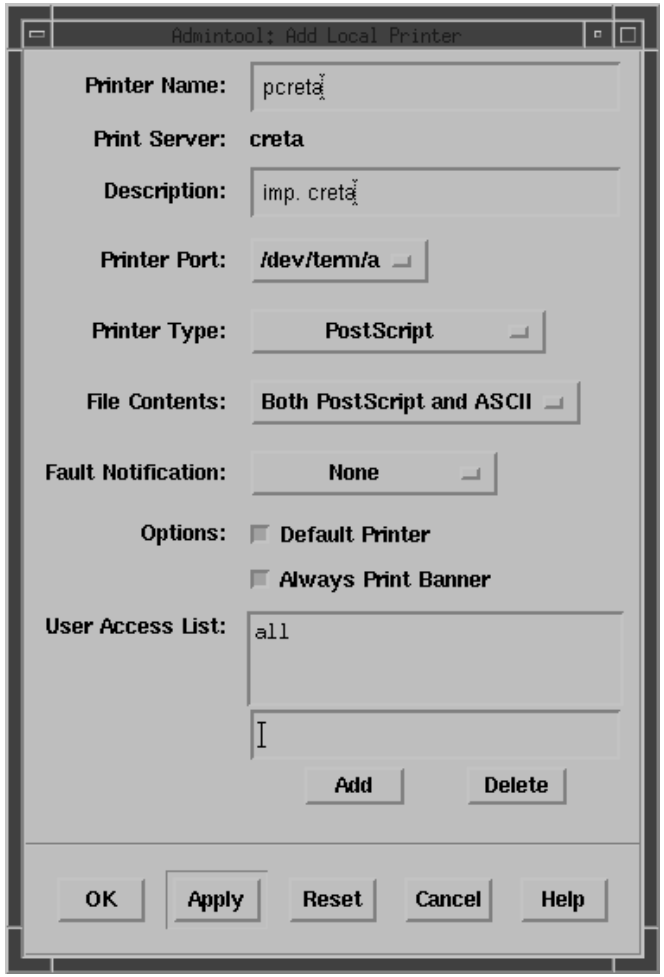
Reiniciar o sistema
#reboot

NOTAS:

11. IMPRESSORAS

11.1. *Definição de Filas de Impressão*

- **Fila local – Admintool**



NOTAS:

- **Fila Local – Comandos lp**

  %lpadmin –p *printername* –*v* device

  %accept *printername*

  %enable *printname*

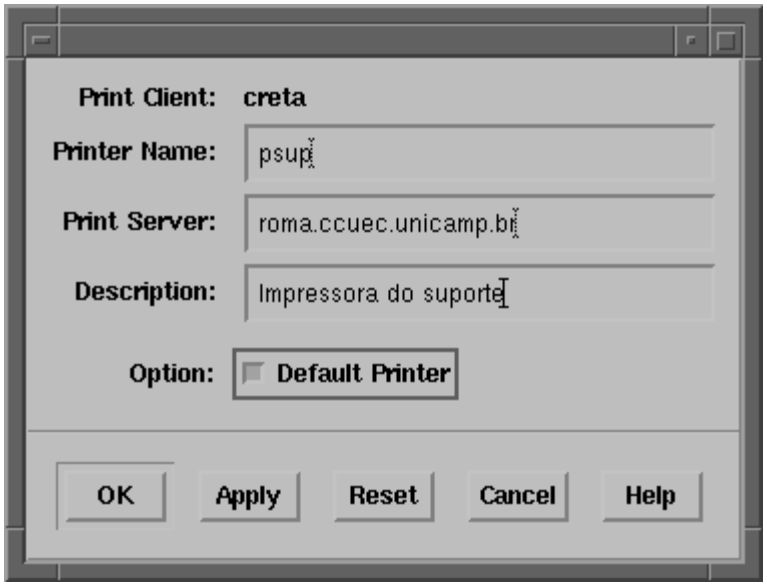
- **Depois de definida a impressora pode tornar-se *default* do sistema usando o seguinte comando:**

  %lpadmin –d *printername*

NOTAS:

O comando *lpadmin* define a fila de impressão e a flag –p define o nome da impressora local, o *accept* habilita a impressora a receber impressões e o comando *enable* habilita a queue a receber impressões

- **Fila remota – Admintool**



NOTAS:

- **Fila Remota – Comandos lp**

%lpsystem –t bsd *servername*

%lpadmin –p *printername* –s *servername*

%lpadmin –p *printername* –T unknown –I any

%accept *printername*

%enable *printername*

NOTAS:

O *lpsystem* define a máquina servidora de impressão.

O comando *lpadmin* define a fila de impressão e a flag –p define o nome da impressora local, o *accept* habilita a impressora a receber impressões e o comando *enable* habilita a queue a receber impressões

11.2. *Comandos de Impressão*

- **Imprimir um arquivo na impressora ln03**
  % lpr -Pln03 file

- **Verificar a fila de impressão na ln03**
  % lpq -Pln03

- **Remover o job 10 da fila da ln03**
  % lprm -Pln03 10

- **Programa de controle de impressoras**

  #lpc
  lpc>

  lpc>abort ln03 (abort current job on print ln03)

  lpc> restart all (restart all printer daemons)

NOTAS:

O comando **lpc** é usado freqüentemente para inicializar uma impressora travada. Em tais casos tente primeiramente:

#lpc restart nome_da_impressora

Se não funcionar tente então:

#lpc stop nome_da_impressora
#lprm primeiro_job_de_impressão
#lpc start nome_da_impressora

No Solaris 2.x o comando de impressão é o **lp**.

11.3. *Destravando Impressoras*

- **Determinar PIDs do processo lpd**

  %ps –ef | grep lpd
  87 ? S 0:02 usr/lib/lpd
  89 ? S 0:05 /usr/lib/lpd
  14459 p6 S 0:00 grep lpd

- **Encerrar os processos**

  %kill –9 87 88

- **Remover o lock da porta da impressora**

  %rm /var/spool/*/lock

- **Remover o socket da impressora**

  %rm devprinter

- **Reinicializar o daemon de impressão**

  %/usr/lib/lpd

NOTAS:

O *daemon* **lpd e inicializado pelo *script* /etc/rc durante o boot do sistema. Ao entrar no ar, o lpd** cria o arquivo lpd.lock no diretório /var/spool. O primeiro **lpd** e inicializado em mod “Master” e “Parent”, e espera serviços através de comandos de impressão tais como **lpr,lpc** e **lprm.**

Quando o trabalho vem de uma impressora, o processo **lpd** pai cria uma cópia de si mesmo. Isto cria um arquivo de *lock* chamado /var/spool/rintername para impedir que os processos **lpd** sejam criados para aquela impressora. Uma vez que a impressora tenha terminado seu trabalho, o processo **lpd** filho morre, mas outro pode ser criado se mais serviço aparecer.

Um sistema que possua muitas impressoras conectadas terá um processo **lpd** Master e um processo **lpd** filho para cada impressora ativa. Uma impressora ativa e aquela que possui jobs sendo impressos. Estes processos filhos são criados e removidos pelo sistema para atender a carga de serviço.

12. NETWORK FILE SYSTEM(NFS)

12.1. *Network File System*

- **Serviço de rede que permite o compartilhamento transparente de sistemas de arquivos ou diretórios entre nós de uma rede**

- **Implementado usando RPC(Remote Procedure Call), cujos protocolos são descritos usando XDR(External Data Representation)**

Notas:

12.2. *Servidor NFS*

- **Exporta sistemas de arquivos para tornar seu uso transparente a aplicações de clientes NFS**
- **Lê ou grava arquivos em resposta a pedidos de clientes NFS**
- **Não mantém informações relativa a arquivos por clientes**
- **Pode servir a clientes NFS ou a outros servidores**
- **Não faz cache de pedidos de write de clientes**

Notas:

12.3. *Cliente NFS*

- **Monta sistema de arquivos compartilhados pelo servidor NFS**
- **Arquivos são lidos ou gravados através de pedido ao servidor NFS**
- **Mantém toda informação a respeito de seus arquivos abertos**
- **Pode utilizar os serviços de vários servidores NFS**
- **Pode se comunicar com servidores NFS através de nós na Ethernet**
- **Realiza cache para gravação**

Notas:

12.4. *Interação Cliente/Servidor*

- O arquivo /etc/dfs/dfstab determina os sistemas de arquivos a serem exportados.
- O servidor inicializa o daemon do NFS (nfsd) e o daemon para mount (mountd)
- O daemon mountd do servidor retorna um indicador (file handle) para o diretório ou sistema de arquivos solicitado
- O file handle do cliente é colocado na tabela de mounts do kernel
- Todas as referências posteriores são passadas ao daemon nfsd rodado no servidor usando o file handle do cliente
- O read ahead/write behind do NFS é gerenciado pelo I/O daemon

NOTAS:

12.5. *Exportação de Sistemas de Arquivos*

- **Opções:**

-ro	**exporta o sistema de arquivo como read-only**
-rw=host1:host2	**read-only para a maioria e read-write para os hosts especificados**
-root=host1:host2	**dá aos hosts listados acesso de root ao sistema de arquivos**
-anon=userid	**atribui a este userid qualquer pedido de um usuário desconhecido**

NOTAS:

O processo de exportação é chamado *sharing*, executado pelo comando *share.*

Para startar o *daemon* nfsd (servidor):
/usr/lib/nfs/nfsd

Para startar o *daemon* mountd (cliente/servidor):
/usr/lib/nfs/mountd

Para compartilhar o filesystem depois de acrescentar o cliente no /etc/dfs/dfstab (servidor):
Share –F nfs –o rw=host /filesystem

Executar o comando share all ou aguradar o próximo boot da máquina.

12.6. *Importação de Sistemas de Arquivos*

- **O comando mount possui duas opções básicas que são rw (read/write) e ro (read only)**
- **A sintaxe das opções para mount nfs variam de sistema para sistema**
- **O arquivo /etc/vfstab contém uma lista dos sistemas de arquivos a serem montados durante o boot**
- **O arquivo que especifica a tabela de mount é o /etc/vfstab**

NOTAS:

As opções de **mount** podem ser especificadas tanto durante a execução do comando ***mount*** ou no arquivo /etc/vfstab. As opções na página ao lado são usadas para especificar como um cliente nfs deseja que um sistema de arquivos seja montado. Quando o cliente nfs emite um pedido de **mount**, as condições sob as quais o sistema de arquivos ou o diretório é exportado pelo servidor nfs tem precedência sobre as opções de **mount** especificadas pelo cliente nfs.

O comando ***mount*** anexa um determinado sistema de arquivos à estrutura de diretórios. O diretório especificado no comando ***mount*** (mount point) deve existir previamente. Se o diretório possuir algum conteúdo antes do ***mount,*** este permanecerá oculto até que o sistema de arquivos seja novamente desmontado.

Para que um cliente nfs efetivamente monte um sistema de arquivos, o servidor nfs precisará estar exportando o mesmo. Se o servidor especificar quem pode importar ou montar o sistema de arquivos solicitado, o cliente precisa estar incluído nesta lista. O servidor nfs também precisa possuir os daemons necessários (na maior parte dos sistemas **rpc.mountd** e **nfsd**).

Para montar (cliente):

```
mount –F nfs servidor:filesystem mountpoint
```

Editar o /etc/vfstab (cliente) para que o filesystem seja montado durante o boot:

```
servidor:/filesystem  servidor:/filesystem  /mountpoint  nfs  3  yes  rw,bg,nosuid
```

nfs – tipo do filesystem montado
3 - momento do boot em que será montado
yes – para ser montado durante o boot
rw - montado como read e write
bg - montado como background
nosuid – não permite que seja rodada shell com setuid bit ligado

12.7. *Comandos Informativos*

```
#showmount obelix
obelix
salvia
fem
apoio


#dfmounts obelix
export list for obelix
/home/obelix/supusr        havana.cmp.unicamp.br
/pub/pub –ro               Everyone


#dfshares obelix
export list for obelix
/home/obelix
/pub/pub
```

NOTAS:

O comando showmount lista todos os clientes que montaram remotamente um sistema de arquivos na máquina especificada. A opção –e exibe a lista de sistemas de arquivos exportados e a opção –d mostra os sistemas de arquivos montados por algum cliente.

13. NETWORK INFORMATION SERVICE(NIS)

13.1. *Network Information Service*

- **Banco de dados com informações na rede**
- **Servidores e clientes**
- **Domínio NIS**

NOTAS:

NIS é um serviço de banco de dados com informações relativas a uma rede de computadores, que pode ser replicado em vários servidores da rede. Cada servidor mestre roda com um processo para o banco de dados e é conhecido como servidor NIS. Os vários servidores propagam os bancos de dados atualizados entre si para assegurar a consistência das informações. Quando o funcionamento está normal, não faz diferença qual servidor respondeu ao pedido, pois a resposta será sempre igual. Isto permite que os múltiplos servidores coexistam em uma mesma rede, proporcionado ao NIS um alto grau de disponibilidade e confiabilidade.

Um domínio NIS consiste de um grupo de nós de uma rede, todos usando o mesmo banco de dados NIS. Domínios NIS podem ser definidos diferentes de domínio Internet e de mail. Um diretório NIS é um diretório dentro de /usr/yp que contém um grupo de mapas. O nome do diretório é o nome do domínio. O nome do domínio é necessário para obter dados de um banco de dados NIS. Cada máquina na rede pertence a um domínio *default* definido pelo comando *domainname* ou no arquivo /etc/defaultdomain. O nome do domínio precisa ser definido em todas as máquinas, tanto servidoras como clientes.

Cada mapa NIS contém um grupo de valores e chaves que são consultadas pelos programas. A maioria dos mapas são derivados de arquivos ASCII encontrados no diretório /etc: hosts, group, passwd e outros.A informação dentro de um mapa NIS é semelhante à contida nos arquivos ASCII, mas é mantida no formato DBM. Os mapas em formato DBM são implementados em dois arquivos com nomes como “mapname.dir” e “mapname.pag”. Por exemplo, *host.bynamer* é armazenada em dois arquivos de nome hostname.pag e hostname.dir. O arquivo terminado em .pag contém as entradas dos mapas.

13.2. *Servidores e Clientes*

- **Servidor Mestre**
    - **Contém os mapas NIS**
    - **Atualiza os mapas NIS**
    - **Propaga os mapas NIS para os servidores escravos**
    - **Fornece mapas NIS para clientes do domínio NIS**
- **Servidor Escravo**
    - **Recebe o grupo adicional de mapas do servidor mestre NIS**
    - **Fornece serviços de NIS aos clientes do domínio NIS**
    - **Contém grupo adicional de mapas NIS**
- **Clientes**

NOTAS:

Um servidor NIS é uma máquina que contém um grupo de mapas que são postos à disposição da rede. O servidor de arquivos não precisa ser o servidor NIS, a menos que ele seja a única máquina na rede que possua discos. Existem dois tipos de servidores NIS, mestres e escravo.

O servidor mestre atualiza os mapas dos servidores escravos. As mudanças são propagadas do servidor mestre para os servidores escravos. Se os mapas NIS são criados ou alterados em servidores escravos no lugar do servidor mestre, o algorítmo do NIS será quebrado. Sempre faça todas as modificações e criações de dados no servidor mestre.

Teoricamente uma máquina pode ser mestre de um mapa e cliente de outro. Recomenda-se com vigor que uma máquina deve ser a mestre de todos os mapas criados dentro de um domínio.

Um cliente NIS é uma máquina que faz uso do serviço NIS e roda processos que solicitem dados dos mapas dos servidores.

13.3. *Ypbind/ypserv*

- **Os clientes NIS inicializam o binding daemon (ypbind) durante o boot**
- **O daemon ypbind emite para a rede um pedido para se ligar a um servidor NIS**
- **A máquina rodando o processo mestre (ypserv) se liga ao cliente que faz o pedido**
- **Todas as consultas feitas pelo cliente serão transmitidas a um daemon ypserv em uma servidora pelo processo ypbind**
- **Se um servidor não responde após um binding bem sucedido, o daemon ypbind retorna ao modo *broadcast* para tentar se conectar a outro servidor**

NOTAS:

Os clientes NIS obtêm informações de um servidor NIS através de um processo de *binding*. O binding é a associação de um domínio com o endereço Internet do servidor e da porta naquele servidor através da qual o processo **ypserv** "ouve" os pedidos de serviço. Esta informação é armazenada no diretório /var/yp/binding, usando um arquivo do tipo domainname.versão.

O programa de *binding* é dirigido por pedido de clientes. Como *bind* é estabelecido por *broadcast,* deve existir ao menos um processo *ypserv* em cada rede.

Binding e *rebindings* são tratados transparentemente pelas rotinas de biblioteca C. Se o processo *ypbind* é incapaz de falar com o processo *ypserv* ao qual se conectou, ele marca o domínio como *unbound* e tenta se ligar ao domínio novamente.

Se o arquivo /var/yp/ypserv.log existe quando o processo ypserv é inicializado, informações de erro serão registradas neste arquivo quando do aparecimento de condições de erro.

Os arquivos /var/yp/binding/domainname.version são criados para acelerar o processo de *binding*. Estes arquivos fazem um cache do último *binding* bem sucedido criado para o domínio; quando um *binding* é solicitado estes arquivos são verificados quanto a sua validade e então usados.

13.4. *Rpc.passwdd*

- **O comando passwd emitido a partir de um cliente NIS se comunica com o daemon rpc.yppasswdd rodando no servidor NIS mestre**

- **Yppasswd**

NOTAS:

/usr/etc/rpc.yppasswdd é o daemon de passwords do NIS.

Observe que este daemon:

- Roda apenas no servidor mestre
- Atende pedidos de troca de passwd
- Pode ser rodado de modo que todos os servidores escravos tenham seus dados atualizados sempre que as passwords forem alteradas

Nas máquinas com SUNOS o comando para alterar passwords em um domínio NIS é o **yppasswd.** O comando age da mesma forma que o comando passwd, mas o yppasswd altera a password no servidor mestre ao invés do arquivo local /etc/passwd. No entanto, quando o usuário entra no sistema através do NIS a senha é mudada automaticamente no banco de dados deste sistema.

13.5. *Arquivos afetados pelo NIS*

- **/etc/passwd**
- **/etc/group**
- **/etc/aliases**
- **/etc/hosts**
- **/etc/hosts.equiv**
- **.rhosts**

NOTAS:

O arquivo /etc/passwd local é consultado em primeiro lugar quando o NIS está rodando. Dados no arquivo passwd local possuem precedência sobre o banco de dados do NIS. As entradas no arquivo passwd local devem ser apenas para o usuário root e usuários locais.

Programas que invocam o arquivo /etc/passwd consultam o arquivo passwd do NIS

O arquivo /etc/group é consultado localmente primeiro. Os dados no arquivo group possuem precedência sobre os mapas do NIS.

Aliases de mail são resolvidos consultando os arquivos /etc/aliases primeiro e então os mapas do NIS. Quando o NIS está rodando, os *aliases* adicionais estão localizados no mapa mail.aliases dos servidores NIS.

Quando o NIS está rodando, o arquivo /etc/hosts é consultado durante o boot. Como ele é consultado durante o boot ele deve conter entradas para o host local e, como para todas as máquinas, o local *loopback localhost*

```
127.0.0.1          localhost
143.106.10.11      obelix
```

Uma vez terminado o boot, apenas o NIS é consultado para resolver endereços de máquinas.

Quando o NIS está rodando, os arquivos hosts.equiv ./rhosts são consultados localmente. Se uma entrada do tipo “+”ou “-“ preceder uma entrada, isoladamente em uma linha, todas as outras entradas são procuradas no banco de dados NIS.

13.6. */etc/passwd*

- **+::::::**
- **+jose**
- **+jose:::::/home/obelix/jose:**
- **Exemplo de arquivo passwd para NIS**

NOTAS:

+::::::
Neste exemplo todas as entradas do banco de dados NIS para *passwords* são válidas nesta máquina. Esta entrada deve ser a última a aparecer no arquivo /etc/passwd.

+jose
O *userid* é válido e seus dados de login devem ser obtidos no banco de dados do NIS.

+jose:::::/home/obelix/jose
jose é um usuário válido para este host. Entretando, os dados que aparecem no arquivo passwd local têm preferência sobre os dados do NIS. Mesmo sendo outro o diretório home indicado pelo NIS, o diretório home final para o usuário é /home/obelix/jose.

O arquivo password ao lado é típico para máquinas clientes de NIS. A última linha indica que um programa que chame o arquivo /etc/passwd (local) irá primeiro consultar o arquivo passwd da própria máquina e em seguida o banco de dados NIS. O arquivo passwd local pode conter entradas como as encontradas ao lado, para alguns usuários em particular, se algumas informações são mantidas localmente, como por exemplo, o diretório home.

13.7. */etc/group*

- **Exemplo de /etc/group:**

```
wheel:*:0:
nogroup:*:65534:
daemon:*:1:
kmem:*:2:
bin:*:3:
tty:*:4:
operator:*:5:
uucp:*:7:
audit:*:9:
staff:*:10:
other:*:20:
+:
```

- **+source**

NOTAS:

+:
Neste exemplo, todos os grupos definidos no NIS são válidos para o host.

+source
O grupo source definido no banco de dados do NIS é válido para este host.

13.8. *Arquivos substituidos pelo NIS*

- **/etc/ethers**
- **/etc/netgroup**
- **/etc/netmasks**
- **/etc/networks**
- **/etc/protocols**
- **/etc/services**
- **/etc/bootparams**

NOTAS:

Estes arquivos nunca são consultados pela máquina local se ela é cliente NIS. Apenas o banco de dados NIS é utilizado para pesquisar informações contidas nestes arquivos, mesmo que a informação solicitada esteja presente no arquivo local.

13.9. *Criação de um servidor NIS mestre*

- **Definir o nome do domínio**

#domainname domínio
(também definir o nome do domínio no arquivo /etc/defaultdomain)

- **Atualizar os arquivos ASCII que serão incluídos no banco de dados do NIS**

- **Criar o banco de dados do NIS**

#cp /var/yp;ypinit –m

- **Inicializar os daemons do NIS**

#/etc/init.d/yp start

NOTAS:

Para instalar os pacotes acima primeiro é necessário o CDROM do pacote de NISkit Server para Solaris. Depois se usa o comando **pkgadd** para instalar os pacotes no sistema: **SUNnisr, SUNWnisu, SUNWnsktr e SUNWnsktu.**

Os comandos necessários ao NIS estarão nos diretórios /usr/sbin e /usr/lib/netsvc/yp.

Se o domínio não foi definido no servidor, use o comando *domainname* para isto. Também definir o nome do domínio no arquivo /etc/defaultdomain.

Se um arquivo passwd adjunto está sendo usado, especifique-o depois da opção –m e indique o diretório onde se encontra, como no exemplo abaixo:
/etc/etc/rpc.uppasswdd /etc/passwd –m passwd DIR=/etc/yp
A opção –m especifica que os mapas do NIS encontrados no diretório /ver/yp devem ser atualizados e retransmitidos aos servidores escravos, se houverem.

Em seguida, antes de começar a instalação, atualize os arquivos ASCII: passwd, hosts, ethers, group, aliases, netgroup, networks, protocols, bootparams e services. Aproveite esse momento para criar os arquivos adjuntos de usuário se necessário.

Rode *ypinit*. A opção –m indica que o banco de dados será atualizado quando mudanças forem feitas nos arquivos ASCII. Se você sabe quais servidores serão os escravos forneça estes nomes quando solicitado. Em seguida inicialize os processos **ypbind, ypserv,** e **rpc.ypppaswdd.**

Quando for preciso atualizar apenas um mapa, pode ser utilizado o comando make, lembrando-se que no caso de passwd o comando é:

Make passwd DIR=/etc/yp

13.10. *Criação de um servidor NIS escravo*

- **Definir o nome do diretório**

#domainname domínio
(também definir o nome do domínio no arquivo /etc/defaultdomain)

- **Inicializar a máquina como cliente**

#ypbind ou #ypinit –c

- **Obtenha uma cópia dos bancos de dados do servidor NIS mestre**

#cp /var/yp
#ypinit –s *mastername*

- **Inicializar os daemons do NIS**

/etc/init.d/yp start

- **Incluir o servidor escravo na lista do servidor mestre, se isto não tiver sido feito quando o mestre foi configurado**

NOTAS:

Os procedimentos para criar-se um servidor NIS escravo, resumem-se aos seguintes:

Definir o domínio NIS

Definir a máquina como cliente

Definir a máquina como servidora NIS escrava

No servidor NIS master , incluir o servidor escravo na lista de servidores.

13.11. *Sincronização de Mapas*

- **Para transferir um mapa do servidor mestre para um servidor escravo**

#ypxfr –h *host* nome_do_mapa

- **Os mapas também podem ser atualizados à partir dos servidores escravos. No Solaris as shells scripts de atualização estão em /usr/lib/netsvc/yp**

- **Ypxfr_1perday altera os mapas:**
  Group
  Protocols
  Networks
  Services
  Ypservers

- **Ypxfr_2perday altera os mapas:**
  Hosts
  Ethers
  Netgroup
  Mail aliases

- **/usr/etc/yp/r_1perhour altera o mapa passwd**

NOTAS:

No Solaris 2.x os comandos e scripts relacionados com a tualização de mapas estão localizados em /usr/lib/netsvc/yp.

Para manter os mapas dos servidores escravos atualizados, o comando ypxfr pode ser rodado nos servidores escravos.Existem *shell scripts* que fazem uso deste comando para transferir alguns mapas. Os nomes dos *scripts* indicam a frequência com que devem ser rodados. Altere os *scripts* para adequa-los à sua instalação.

Estes *scripts* são normalmente rodados pelo *crontab* da área do root. Estes horários devem ser intercalados com os de outros servidores para evitar sobrecarregar o servidor mestre.

As tentativas de transferências de mapas pelo comando *ypxfr* e seus resultados são registradas no arquivo ypxfr.log.

Este arquivo pode ser criado utilizando-se o comando:

#touch /var/yp/ypxfr.log

Para desativar o registro das atividades do comando *ypxfr* apague o arquivo ypxfr.log.

13.12. *Acrescentando Clientes*

- **Altere o nome do domínio**

#domainname domain
(definir o nome do domínio NIS no arquivo /etc/defaultdomain)

- **Inicialize o daemon *ypbind***

#ypbind ou ypinit –c

NOTAS:

13.13. *Atualização do Banco de Dados do NIS*

- **Todas as mudanças no banco de dados são feitas a partir do servidor mestre do NIS**
- **Altere os arquivos ASCII necessários**
- **Vá para o diretório NIS**
- **Reconstrua os mapas NIS que foram alterados e propague-os para os servidores escravos**

NOTAS:

Por exemplo: o administrador altera o arquivo ASCII /etc/passwd no servidor mestre do NIS e roda o comando **make** que irá atualizar o mapa do arquivo /etc/passwd. A versão atualizada do mapa é enviada para os servidores escravos:

Exemplo (apenas no servidor mestre do NIS)
Master# vipw ou useradd (altera o arquivo passwd e shadow)
Master# cd /var/yp
Master# make

O comando **make** reconstrói apenas os mapas que necessitam de mudanças. Quando as mudanças estiverem completas, o comando **make** invoca o comando **yppush** para propagar os mapas para os servidores escravos.

13.14. *Comandos Informativos do NIS*

- **Ypcat**
- **Ypwhich**
- **Ypwhich –m**
- **Ypwhich clientname**
- **Yppush**
- **Ypset *host***
- **Ypset *map***

NOTAS:

- /usr/bin/ypwhich
  mostra quem é o servidor do NIS
- /usr/bin/ypwhich –m
  mostra quem é o mestre de cada mapa
- /usr/bin/ypwhich *clientename*
  mostra qual servidor está prestando serviço de NIS para uma máquina cliente
- /usr/etc/yp/yppush
  roda no servidor mestre e copia uma nova versão de um mapa NIS do servidor mestre para os servidores escravos
- /usr/etc/yp/yppoll –h *host*
  pergunta ao processo **ypserv** no host indicado acerca dos parâmetros do mapa: o número de ordem e qual host é o servidor mestre do NIS para o mapa especificado
- /usr/bin/ypcat map
  exibe o conteúdo de um mapa do NIS

14. SEGURANÇA

14.1. *INTRODUÇÃO*

- O Sistema operacional Unix não foi projetado com preocupações relativas à segurança
- Sistemas instalados com pouco ou nenhum mecanismo de segurança
- O unix foi um sistema desenvolvido por programadores para programadores
- A conexão em rede agravou o problema
- A difusão do uso de sistemas Unix está tornando estes sistemas mais seguros
- As características básicas de segurança do UNIX estão ligadas às contas de usuários, aos sistemas de arquivos e à conetividade.

Notas:

14.2. *Segurança das Contas*

- **Senhas**
  - **Seleção adequada**
  - **Política para escolha**
  - **Verificação periódica da segurança**
- **Data de expiração**
- **Negar contas *guest***
- **Contas sem senha**
- **Contas de grupo**
- **NIS**

NOTAS:

A segurança de sistemas Unix pode ser dividida em três áreas principais. A segurança de contas consiste basicamente impedir que usuários não autorizados acessem o sistema; segurança de rede consiste em não permitir que os pacotes sejam capturados por equipamentos ou programas instalados sem autorização e que os dados importantes trafeguem criptografados; a segurança dos sistema de arquivos, consiste em impedir que usuários não autorizados, tanto do sistema, quanto invasores, consigam acesso aos dados armazenados no sistema.

Contas:

Uma das maneiras mais fáceis de um intruso ganhar acesso a um sistema, é descobrindo a senha da conta de algum usuário. Isto não é difícil de ocorrer já que muitas instalações não removem o acesso de pessoas que já deixaram a organização, não verificam a segurança das contas em vigor, com *passwords* fáceis de serem descobertas.

14.3. *Segurança do Sistema de Arquivos*

- **Permissões dos arquivos**
  - **Cada diretório ou arquivo possui três grupos de permissões:**
    - **dono do arquivo**
    - **grupo do dono do arquivo**
    - **demais usuários**
- **Cada grupo de permissões pode especificar os seguintes atributos:**
  - ***read***
  - **write**
  - **execute**
  - **setuid (4xxx)**
  - **setgid (2xxx)**
  - **sticky (1xxx)**

NOTAS:

A última linha de defesa contra invasores são as permissões oferecidas pelo sistema. Cada arquivo ou diretório possui três grupos de bits de permissão a ele associados: um grupo para o usuário ao qual o arquivo pertence, um grupo para os usuários do grupo ao qual o arquivo está associado, e um grupo para todos os demais usuários. Cada grupo contém três grupos de bits que controlam:

- Acesso de leitura (read):

Se este bit estiver ligado, o arquivo ou diretório possui acesso de leitura. Em se tratando de um diretório, esta permissão permite a um usuário ver o conteúdo de um diretório mas não acessá-lo.

- Acesso de gravação (write):

Se este bit estiver ligado, o arquivo ou diretório possui acesso de leitura. Em se tratando de um diretório, esta permissão de escrita implica a capacidade de criar, apagar, ou renomear arquivos. Note que a permissão para remover um arquivo não é determinada pelas permissões do arquivo, mas sim pelas permissões do diretório que o contém.

- Acesso de execução (execute):

Se este bit estiver ligado, o arquivo ou diretório pode ser executado (pesquisado). No caso de um diretório, a permissão de execução implica que os arquivos nele contidos podem ser acessados.

Além destas permissões, existe um quarto bit que se ligado atribui as seguintes propriedades aos arquivos e diretórios:

- *setuid:*

Este bit, se ativado no grupo de permissões do dono do arquivo, indica que todos os que executarem este programa o estarão fazendo com os privilégios do proprietário do arquivo. Por exemplo, o programa *sendmail* é setuid *root*., o que lhe permite gravar mensagens do sistema, o que é vedado a usuários normais. Este bit não possui significado em arquivos não executáveis.

- *setgid*:

Este bit atua da mesma forma que o setuid, com a diferença que o programa será executado com as permissões do grupo ao qual pertence.

- *stcky*:

O *sticky* informa ao sistema operacional para atuar diferentemente com relação à imagem executável do arquivo. É remanescente de versões antigas do Unix e possui pouco uso hoje. Diretórios que possuem modo 777 e têm este bit ligado não podem ser removidos (/tmp por exemplo).

- **Nunca utilizar ou permitir que se utilize em qualquer sistema Unix shell scripts com o *stuid bit* ligado**

- **Sticky bit em diretórios**

- **setgid em diretórios**

- **Usar Umask para que todos arquivos criados fiquem com a permissão desejada.**

#umask 022 (resulta arquivos regulares com 644 → rw-r—r-)

- **Arquivos encriptados**

- **Dispositivo**

NOTAS:

Shell scripts que possuem os bits setuid ligados não são seguros, não importa quantas precauções tenham sido tomadas ao escrevê-los. Tais scripts nunca devem ser permitidos em qualquer sistema Unix.

Nas versões mais recentes de sistema Unix foi acrescentado um novo significado ao sticky bit quando aplicado à diretórios. Neste caso, os usuários não podem apagar e renomear arquivos pertencentes a outros usuários. Isto é tipicamente útil no caso diretórios com /tmp. Normalmente o diretório /tmp possui acesso universal para gravação, permitindo que qualquer usuário remova arquivos pertencentes a outro usuário. Ativando o sticky bit no arquivo /tmp, os usuários podem remover apenas seus próprios arquivos. Para ativar o sticky bit em um diretório, use o comando:
#chmod o+t *diretório*

Ao se criar um arquivo normalmente todas as permissões são ativadas. Como isto raramente é o desejado, o valor do *umask* é usado para modificar o grupo de permissões com as quais um arquivo é criado. Ou seja, da mesma forma com o comando *chmod* especifica quais bits devem ser ligados, o comando *umask* especifica quais bits devem ser desligados.

O comando *umask* é especificado nos arquivos .cshrc ou .profile, dependendo da shell utilizada. A conta root deve ter a *umask* definida como 022 , para impedir a criação acidental de arquivos pertencentes ao superusuário com permissão 777.

A segurança de dispositivo é uma questão importante em sistemas Unix. Arquivos de dispositivo são utilizados por vários programas para acessar dados nos dispositivos ou na memória. Se estes dispositivos não estão devidamente protegidos, o sistema está vulnerável a ataques. A lista completa de dispositivos é muito grande e varia de sistema para sistema. Em linhas gerais, as seguintes normas se aplicam:

- Os arquivos **/dev/kmem**, **/dev/mem** e **/dev/drum** não devem ter permissão de leitura universal.
- Os dispositivos de disco, tais como **/dev/sd0a**, **/dev/rx1b**, etc., devem pertencer ao usuário root e group operator, e devem possuir modo 640.
- Como muito poucas exceções, todos os outros dispositivos devem pertencer ao usuário root. Uma destas exceções são os terminais, que pertencem ao usuário que o estiver utilizado no momento. Ao desconectar-se, o terminal volta a pertencer ao root.

14.4. *Segurança da Rede*

- ***Trusted hosts***
  - **/etc/hosts.equiv**
  - **.rhosts**
- **Terminais seguros**
- ***Network File System* (NFS)**
  - **/etc/exports**
  - **/etc/netgroup**
  - **Restrição de acesso para o superusuário (root)**
- ***File Transfer Protocol* (FTP)**
- ***tftp***

NOTAS:

- **Mail**
  - **Cuidado na criação de alias que enviem mensagens para programas**
  - **Verificar que o programa sendmail não autoriza o comando debug**
- **Cuidado na instalação de programas públicos**
- **Finger**
- **Modems e servidores de terminais (ttys)**
- **Firewalls**

NOTAS:

14.5. *Monitoramento da Segurança*

- **Segurança de contas**
  - **lastlog**
  - **utmp**
  - **wtmp**
  - **acct**
- **Segurança da rede**
  - **syslog**
  - **showmount**

NOTAS:

Uma das tarefas do administrador de sistemas é a monitoração da segurança. Esta tarefa envolve o exame de arquivos de log para detectar acessos não autorizados, bem como a monitoração de falhas de segurança.

As contas devem ser monitoradas periodicamente de modo a verificar dois eventos: usuários que logam quando não devem (por exemplo, tarde da noite ou quando estão de férias) e usuários executando comandos que normalmente não deveriam usar.

O arquivo **/usr/adm/lastlog** registra o login mais recente a cada usuário do sistema. A mensagem impressa no terminal a cada vez que um usuário loga usa a data armazenada no arquivo **lastlog**:

Last login: sat Mar 10 10:50:48 from turing.unicamp.br

A data do último login relatada pelo comando **finger** também usa estes dados. Os usuários devem ser alertados a inspensionar esta data para certificarem-se de que não foi efetuado nenhum acesso não autorizado às contas e, caso positivo, a alertar o administrador de sistemas para o ocorrido.

O arquivo **/etc/utmp** é usado para registrar quem está logado no sistema no momento. Este arquivo pode ser exibido através do comando **who**.

O arquivo /usr/adm/wtmp registra cada as datas de login e logout de cada usuário. Este arquivo também pode ser examinado através do comando **who** (who /usr/adm/wtmp).

O arquivo wtmp pode também ser examinado manualmente através do comando **last**. Este comando ordena as entradas no arquivo, casando os tempos de login e logout. Se invocado sem argumentos, o comando **last** exibe toda a informação contida no arquivo.

O arquivo **acct** registra a execução de cada comando no sistema, quem o executou e quanto tempo gastou. O arquivo **acct** pode ser examinado através do comando **lastcomm**.

O **syslog** é um mecanismo que permite que qualquer comando registre mensagens de erro e informativas na console do sistema e/ou em um arquivo. Normalmente mensagens de erro são gravadas no arquivo **/var/adm/messages** juntamente com a data e hora em foram enviadas pelo programa que as gerou.

- **Segurança do sistema de arquivos**
  - **find**
    - **arquivos *setuid/setgid***
    - **arquivos sem dono**
    - **.rhosts**
  - **checklists**
  - **backups**

NOTAS:

Verificar falhas de segurança no sistema de arquivos é outra tarefa importante do administrador. Primeiramente devem ser identificados os arquivos que podem ser alterados por usuários não autorizados, arquivos que podem involuntariamente dar permissões excessivas a usuários e arquivos que podem involuntariamente dar permissões excessivas a usuários e arquivos que possam fornecer acesso a invasores. É importante também monitorar modificações no sistema de arquivos e possuir mecanismos que permitam a volta do sistema ao estado original.

O comando **find** é um comando de propósito geral para pesquisar o sistema de arquivos. A programação do comando **find** listada a seguir localiza todos os arquivos do sistema com os bits setuid e setgid ligados. A saída deste comando deve ser analisada para determinar se não existe algum arquivo suspeito na lista.

```
#find / -type f  -a \(-perm 0400 -o -perm 0200 \) -print
```

Usando as opções que se seguem, o comando **find** identifica os arquivos com permissão de escrita universal.

```
#find / -perm -2 -print
```

Para identificar arquivos que não pertencem a nenhum usuário ou nenhum grupo:

```
#find / -nouser -o nogroup -print
```

Imediatamente após a instalação de um sistema, deve-se gerar um arquivo que liste a configuração inicial dos arquivos do sistema:

```
#lls -aslgR /bin /etc /usr > Masterchecklist
```

Este arquivo contém uma lista completa de todos os arquivos nestes diretórios. As linhas referentes a arquivos que mudem freqüentemente devem ser removidas do arquivo. O **Masterchecklist** deve ser guardado em um local seguro para evitar adulterações. Para pesquisar alterações no sistema de arquivos, execute o comando acima e compare com o arquivo mestre:

```
#diff Masterchecklist Currentlist
```

Outro aspecto muito importante é a realização de backups freqüentes do sistema de arquivo. *Backups* não apenas protegem contra falhas de hardware como contra deleções acidentais.

14.6. *Comandos que podem auxiliar na segurança*

Além dos programas de monitoração existem comandos simples do Unix que podem ser úteis na detecção de violações de segurança. O administrador deve se familiarizar com os comandos executados normalmente no sistema de forma a ser capaz de identificar suspeitas

- **ps**
- **who**
- **w**
- **ls**

NOTAS:

14.7. *Ferramentas Úteis*

- **npasswd (passwords mais seguras)**
- **crack (verifica passwords)**
- **cops (identifica riscos em UNIX)**
- **tripwire (monitora mudanças em binários)**
- **tcpwapper (controla acesso a rede)**
- **kerberos (sistema de identificação)**

NOTAS:

15. LEITURA RECOMENDADA

15.1. *Leitura Recomendada*

- **Manuais originais de cada Sistema Operacional e de cada equipamento**
  - **Sun System Administration**

- **Evi Nemeth e Garth Snider e Scott Seebass, "Unix System Administration Handbook", Pretice Hall, 1989.**

- **David A. Curry "Improving the secure of your System", ftp.unicamp.br: /pub/security/info/security.doc.tar.z**

NOTAS:

16. LINKS RECOMENDADOS

16.1. *Links Recomendados*

- **http://**
  - **sunsolve1.sun.com**
  - **sunsolve.sun.com/sunsolve/pubpatches/patches.html**
  - **www.wins.uva.nl/pub/solaris/solaris2.html**
  - **www.latech.edu/sunman.html**
  - **clamserv.rutgers.edu:8888**
  - **www.Latech.edu/sunman.html**
  - **smc.vnet.net/solaris-2.5.html**

Notas:

17. LISTAS RECOMENDADAS

17.1. *Listas Recomendadas*

- **Majordomo@sunmanagers.ececs.uc.edu**
- **dicas-l@unicamp.br**
- **O newsgroup : comp.sys.sun.adm**
- **Freebsd@sbq.org.br**
- **Linux-br@unicamp.br**

NOTAS: