LINUX

Introdução

O Linux:

1. É um clone UNIX de distribuição livre para PCs baseados em μPs 386/486/Pentium.
2. É uma implementação independente da especificação POSIX, com a qual todas as versões do UNIX padrão (true UNIX) estão convencionadas.
3. Foi primeiramente desenvolvido para PCs baseados em 386/486/Pentium, mas atualmente também roda em computadores Alpha da DEC, Sparcs da SUN, máquinas M68000 (semelhantes a Atari e Amiga), MIPS e PowerPCs.
4. Foi escrito inteiramente do nada, não há código proprietário em seu interior.
5. Está disponível na forma de código objeto, bem como em código fonte.
6. Pode ser livremente distribuído nos termos da GNU General Public License (veja apêndice).
7. Possui todos as características que você pode esperar de um UNIX moderno, incluindo:

- Multitarefa real
- Memória virtual
- Biblioteca compartilhada
- "Demand loading"
- Gerenciamento de memória próprio
- Executáveis "copy-on-write" compartilhados
- Rede TCP/IP (incluindo SLIP/PPP/ISDN)
- X Windows

A maioria dos programas rodando em Linux são freeware genéricos para UNIX, muitos provenientes do projeto GNU. Muitas pessoas tem executado benchmarks em sistemas Linux rodando em 80486, e tem achado o Linux comparável com workstations médias da Sun e da Digital. Ele está disponível através da Internet por meio de centenas de sites FTP.

O Linux está sendo usado hoje em dia por centenas e centenas de pessoas pelo mundo. Está sendo usado para desenvolvimento de softwares, networking (intra-office e Internet), e como plataforma de usuário final. O Linux tem se tornado uma alternativa efetiva de custo em relação aos caros sistemas UNIX existentes. Um exemplo de pacote de distribuição do Linux mais populares é distribuido pela InfoMagic (http://www.infomagic.com, e-mail info@infomagic.com), a versão LINUX Developer's Resource CD-ROM, de dezembro de 1996, contém 6 CD-ROMs, seu conteúdo sucinto é :

- Versão Red Hat 4.0 (instalando kernel 2.0.18)
- Versão Slackware 3.1 (Slackware 96 - instalando kernel 2.0)
- Versão Debian GNU/Linux 1.2
- X-Windows - Xfree86 version 3.2
- Arquivos Linux de tsx-11.mit.edu e sunsite.unc.edu
- Arquivos GNU de prep.ai.mit.edu
- Documnetação completa on-line & HOWTO's (Guia de Instalação e Guia do Administrador da Rede, em inglês)
- Softwares demostração comerciais como : BRU, dbMan, StarOffice, Cockpit, Flagship, Smartware, GP Modula-2, Pathfinder, Scriptum, etc.

Historia do Linux

O Kernel do Linux foi, originalmente, escrito por Linus Torvalds do Departamento de Ciência da Computação da Universidades de Helsinki, Finlândia, com a ajuda de vários programadores voluntários através da Internet.

Linus Torvalds iniciou cortando (hacking) o kernel como um projeto particular, inspirado em seu interesse no Minix, um pequeno sistema UNIX desenvolvido por Andy Tannenbaum. Ele se limitou a criar, em suas próprias palavras, "um Minix melhor que o Minix" ("a better Minix than Minix"). E depois de algum tempo de trabalho em seu projeto, sozinho, ele enviou a seguinte mensagem para comp.os.minix:

> Você suspira por melhores dias do Minix-1.1, quando homens serão homens e escreverão seus próprios "device drivers" ? Você está sem um bom projeto e esta morrendo por colocar as mãos em um S.O. no qual você possa modificar de acordo com suas necessidades ? Você está achando frustrante quando tudo trabalha em Minix ? Chega de atravessar noites para obter programas que trabalhem correto ? Então esta mensagem pode ser exatamente para você.
>
> Como eu mencionei a um mês atrás, estou trabalhando em uma versão independente de um S.O. similar ao Minix para computadores AT-386. Ele está, finalmente, próximo do estágio em que poderá ser utilizado (embora possa não ser o que você esteja esperando), e eu estou disposto a colocar os fontes para ampla distribuição. Ele está na versão 0.02... contudo eu tive sucesso rodando bash, gcc, gnu-make, gnu-sed, compressão, etc. nele.

No dia 5 de outubro de 1991 Linus Torvalds anunciou a primeira versão "oficial" do Linux, versão 0.02. Desde então muitos programadores têm respondido ao seu chamado, e têm ajudado a fazer do Linux o Sistema Operacional que é hoje.

Ultimas versões do kernel do Linux

```
Release v1.0
        1.0.9
        Data: Sat Apr 16 21:18:02 UTC 1994

Release v1.1
        1.1.95
        Data: Thu Mar 2 07:47:10 UTC 1995

Release v1.2
        1.2.13
        Data: Wed Aug 2 12:54:12 UTC 1995

Release v1.3
        pre2.0.14
           Data: Thu Jun 6 19:30:56 UTC 1996
Release v2.0
        2.0.28
        Data: Tue Jan 14 12:33:26 UTC 1997
        ftp://ftp.cs.Helsinki.FI/pub/Software/Linux/Kernel/v2.0/Linux-2.0.28.tar.gz
Release v2.1
        2.1.23
        Data: Sun Jan 26 14:12:18 UTC 1997
        ftp://ftp.cs.Helsinki.FI/pub/Software/Linux/Kernel/v2.1/Linux-2.1.23.tar.gz
```

Gerência de Processos

Considerações Iniciais

Para explicarmos como o Linux gerência processos, faremos considerações iniciais sobre o código fonte do kernel do Linux (onde encontramos a implementação da Gerência de Processos) e a inicialização "boot" do sistema.

Neste tópico tentaremos explicar, de uma maneira ordenada o código fonte do Linux, tentando conseguir um bom entendimento sobre como o código fonte está situado e como as características mais relevantes do UNIX foram implementadas. O objetivo é ajuda-lo a se familiarizar com o projeto geral do Linux. Então, vamos começar por onde o Linux começa: seu sistema de boot.

Um bom entendimento da linguagem C é necessário para entender este material, assim como familiaridade com conceitos de UNIX e arquitetura dos PCs. Porém, nenhum código C aparecerá neste material, mas referencias de onde podem ser encontrados.

Qualquer referencia "pathname" à arquivos tem como ponto de partida a arvore principal de fontes, usualmente /usr/src/Linux.

A maioria das informações reportadas aqui tem como referencia o código fonte do Linux versão 1.0. Referencias a versões posteriores conterão o símbolo **novo**.

Caso o símbolo não estiver presente, significa que não houveram modificações após as versões 1.0.9-1.1.76.

mais ocasionalmente um parágrafo como este ocorrerá no texto. Indicando onde podem ser obtidas mais informações sobre o assunto corrente (geralmente o código fonte).

Inicialização ("boot" do sistema)

Quando o PC é ligado, o processador 80x86 encontra-se em modo real e executa o código contido no endereço 0xFFFF0, que corresponde a um endereço ROM-BIOS. O BIOS do PC realiza alguns testes no sistema e inicializa o vetor de interrupções no endereço físico 0. Depois disto ele carrega o primeiro setor do device bootavel em 0x7C00, e passa a execução para este endereço. O device é, usualmente, o disquete ou o disco rígido. A descrição anterior é um tanto simplificada, mas é tudo que se necessita para entender o trabalho inicial do kernel.

A primeiríssima parte do kernel Linux está escrito em linguagem assembly 8086 (boot/bootsect.S). Quando é executado, ele se move para o endereço absoluto 0x90000, carrega os próximos 2 kBytes de código do device de boot até o endereço 0x90200, e o resto do kernel para o endereço 0x10000. A mensagem "Loading..." é apresentada durante o carregamento do sistema. O controle é, então passado para o código contido em boot/Setup.S, outro código assembly de modo real.

A parte de "setup" identifica algumas características do sistema (hardware) e o tipo da placa VGA. Se requerido, pede ao usuário para escolher o modo do vídeo da console. E, então, move todo o sistema do endereço 0x10000 para o endereço 0x1000, passa para o modo protegido e passa o controle para o resto do sistema (endereço 0x1000).

O próximo passo é a descompressão do kernel. O código em 0x1000 vem de zBoot/head.S que inicializa os registradores e invoca decompress_kernel(), o qual é composto por zBoot/inflate.c, zBoot/unzip.c e zBoot/misc.c. O dado "descompresso" vai para o endereço 0x100000 (1 Mega), e esta é a principal razão do por que o Linux não pode rodar com menos de 2 Megas de RAM.

mais O encapsulamento do kernel em um arquivo gzip é realizado por Makefile e utilitários no diretório zBoot. São arquivos interessantes para se dar uma olhada.

novo A versão 1.1.75 moveu os diretórios boot e zBoot para arch/i386/boot. Esta modificação pretendeu possibilitar a construção de "kernel verdadeiro" para diferentes arquiteturas.

O código "descompresso" é executado a partir do endereço 0x1010000 , onde todo o setup 32-bit esta lotado: IDT, GDT e LDT são carregados, o processador e o co-processador são identificados, a rotina start_kernel é invocada. Os arquivos fonte das operações acima estão em boot/head.S. Este, talvez, seja o código mais difícil em todo o kernel do Linux.

Note que se algum erro ocorrer durante alguns dos passos precedentes, o computador irá travar. O sistema operacional não pode manipular erros enquanto não estiver totalmente operante.

start_kernel() reside em init/main.c. Tode de agora em diante esta codificado em linguagem C, exceto gerência de interrupções e chamadas de sistemas (Bem, a maior parte das macros possuem códigos assembly embutidos, também).

Depois dos procedimentos com todas as questões iniciais, start_kernel() inicializa todas as partes do kernel, especificamente:

- Inicializa a memória e chama paging_init().
- Inicializa os traps, canais IRQ e scheduling.
- Se requerido, aloja um profiling buffer.
- Inicializa todos device drives e buffers de discos, bem como outras partes menores.
- Regula o delay loop (calcula o numero "BogoMips").
- Checa se a interrupção 16 está trabalhando com o co-processador.

Finalmente, o kernel está pronto para move_to_user_mode(), em seguida fork (bifurca) o processo de inicialização, cujos códigos estão no mesmo arquivo fonte. E o processo número 0, também chamado idle task (tarefa preguiçosa), se mantém rodando em um loop infinito.

O processo de inicialização tenta executar /etc/init, ou /bin/init, ou /sbin/init.

Se nenhum deles tem sucesso, o código se desvia para "/bin/sh /etc/rc" e cria um root shell no primeiro terminal (console). Este código é remanescente do Linux 0.01, quando o S.O. era feito para um kernel stand-alone, e não havia processo de login.

Depois de exec() o programa de inicialização de um dos lugares padrão (deve haver um deles), o kernel não tem controle direto sobre o fluxo do programa. Sua função, de agora em diante, é prover processos através de chamadas ao sistema (system calls), assim como prover eventos para serviços assíncronos (como uma interrupção do hardware). A multitarefa está inicializada, e inicializará o gerenciamento de acesso a multiusuários, através do fork() e processos de login.

Estando o kernel carregado e provendo serviço, vamos prosseguir dando uma olhada nesses serviços ("system calls").

Gerência de processo pelo kernel

Do ponto de vista do kernel, um processo é uma entrada na tabela de processos. Nada mais.

A tabela de processos, então, é uma das mais importantes estruturas de dados no sistema, conjuntamente com a tabela de gerenciamento de memória e o buffer cache. O item individual na tabela de processos é a estrutura task_struct, definida em include/Linux/sched.h. Com a task_struct, tanto informações de baixo quanto de alto nível, são mantidas – variando da cópia de alguns registradores de hardware até o inode do diretório de trabalho para o processo.

A tabela de processos é tanto um array quanto uma lista duplamente ligada, como uma árvore. A implementação física é um array estático de ponteiros, cujo tamanho é NR_TASKS, uma constante definida em include/Linux/tasks.h, e cada estrutura reside em uma pagina de memória reservada. A estrutura da lista está entre os ponteiros next_task e prev_task, a estrutura em arvore é um tanto complexa, e não será descrita aqui. Voce pode desejar mudar NR_TASKS do seu valor default (que é 128), mas esteja certo de que há dependências, e será necessário recompilar todos os arquivos fonte envolvidos.

Depois do boot, o kernel está sempre trabalhando em um dos processos, e a variável global "current", um ponteiro para um item da task_struct, é usado para

guardar o processo que está rodando. A variável "current" só é mudada pelo scheduler, em kernel/sched.c. Quando, porém, todos os processos necessitarem estar looked, a macro for_each_task é usada. Isto é consideravelmente mais rápido que uma procura seqüencial no array.

Um processo está sempre rodando em ou em "modo usuário" ou em "modo kernel". O corpo principal de um programa de usuário é executado em modo usuário e chamadas a sistema são executados em modo kernel. A pilha usada pelos processos netes dois modos de execução são diferentes – um seguimento de pilha convencional é usado para o modo usuário, enquanto uma pilha de tamanho fixo (uma página, cujo processo é dono) é usada no modo kernel. A página de pilha para o modo kernel nunca é swapped out, porque ela pode estar disponível sempre que um system call é introduzido.

Chamadas a sistema (System calls), no kernel do Linux, são como funções da linguagem C, seu nome "oficial" esta prefixado por "sys_". Uma chamada a sistema de nome, por exemplo, burnout invoca a função de kernel sys_burnout().

mais O mecanismo de chamadas a sistema (System calls) está descrito no capítulo 3 do Linux Kernel Hackers' Guide http://www.redhat.com:8080/HyperNews/get/khg.html. Uma olhada em for_each_task e SET_LINKS, em include/Linux/sched.h pode ajudar a entender a lista e a estrutura de árvore da tabela de processos.

Criando e destruindo processos

Um sistema UNIX cria um processo através da chamada a sistema fork(), e o seu término é executado por exit(). A implementação do Linux para eles reside em kernel/fork.c e kernel/exit.c.

Executar o "Forking" é fácil, fork.c é curto e de fácil leitura. Sua principal tarefa é suprir a estrutura de dados para o novo processo. Passos relevantes nesse processo são:

- Criar uma página livre para dar suporte à task_struct
- Encontrar um process slot livre (find_empty_process())
- Criar uma outra página livre para o kernel_stack_page
- Copiar a LTD do processo pai para o processo filho
- Duplicar o mmap (Memory map - memoria virtual) do processo pai

sys_fork() também gerencia descritores de arquivos e inodes.

novo A versão 1.0 do kernel possui algum vestígio de suporte ao "threading" (trabalho ou processo em paralelo), e a chamada a sistema fork() apresenta algumas alusões à ele.

A morte de um processo é difícil, porque o processo pai necessita ser notificado sobre qualquer filhos que existam (ou deixem de existir). Além disso, um processo pode ser morto (kill()) por outro processo (isto é um aspecto do UNIX). O arquivo exit.c é, portanto, a casa do sys_kill() e de variados aspectos de sys_wait(), em acréscimo à sys_exit().

O código pertencente à exit.c não é descrito aqui - ele não é tão interessante. Ele trabalha com uma quantidade de detalhes para manter o sistema em um estado consistente. O POSIX "standard", por conseguinte, é dependente de sinais (flags), e tinha que trabalhar com eles.

3.4 - Executando Processos

Depois de executar o fork(), duas copias do mesmo programa estão rodando. Uma delas usualmente executa - exec() - outro programa. A chamada a sistema exec() deve localizar a imagem binária do arquivo executável, carrega-lo e executa-lo. "Carrega-lo" não significa, necessáriamente, copiar na memória a imagem binária do arquivo, para que, assim, o Linux possa atender a demanda de programas a serem executados.

A implementação Linux do exec() suporta formatos binários diferentes. Isto é dotado através da estrutura Linux_binfmt, a qual embute dois ponteiros para

funções - um para carregar o executável e o outro para carregar a "library" associada, cada formato binário deve conter, portanto, o executável e sua "library".

O sistema UNIX prove, ao programador, seis formas para a função exec(). Quase todos podem ser implementados como uma "library" de funções, e o kernel do Linux implementa sys_execve() independentemente das providas pelo UNIX. Ele executa uma única tarefa: carregar o cabeçalho do executável, e tenta executa-lo. Se os dois primeiros bytes são "#!", então a primeira linha é ignorada e um interpretador é invocado, caso contrário o formato binário, registrado, é executado seqüencialmente.

O formato nativo do Linux é suportado diretamente por fs/exec.c, e as funções relevantes são load_aout_binary e load_aout_library. Assim como para os binários a função de carregamento "a.out" é invocada, e a função mmap() (memory map - memória virtual) aloca espaço em disco (no caso da memória real estar cheia) para o processo, ou invoca read_exec(), caso haja espaço em memória. "The former way uses the Linux demand loading mechanism to fault-in program pages when they're accessed, while the latter way is used when memory mapping is not supported by the host filesystem (for example the "msdos" filesystem)".

novo A partir da versão 1.1 do kernel, o Linux embutiu um sistema de arquivos (filesystem) revisado do msdos, que suporta mmap() (memory map - memória virtual). Além disso a estrutura Linux_binfmt é uma "lista ligada" e não um array, para permitir carregar um novo formato binário como um módulo do kernel. Finalmente a estrutura, por si mesma, foi estendida para acessar rotinas com o formato relativo à core-dump.

4 - Gerência de Memória

4.1 - Gerenciamento de Memória do Linux *(LMM)*

A execução do LMM (Linux Memory Manager) exige uma estratégia de paginação com uma copy-on-write confiando nas 386 páginas auxiliares. Um processo alcança suas tabelas de páginas de seu parent (durante um fork) com as entradas marcadas como read-only ou trocado. Então, se o processo tenta escrever para este espaço de memória e a página é uma copy on write page, isto é copiado e a página marcada read-write. Um exec ( ) resulta na leitura de uma página ou mais do executável. O processo então erra em qualquer outra página que precisar.

Cada processo tem uma tabela de página que significa que pode acessar 1 Kb de tabela de página indicando para 1 Kb de 4 Kb, páginas que é 4 Gb de mémoria. Um diretório de página do processo é iniciado durante um Fork por copy-page-tables. O processo inativo tem seu diretório de página inicializado durante a sequência de inicialização.

Cada processo usuário tem uma tabela descritória local que contém um código de segmento e um segmento de dados. Estes segmentos usuários extendem de 0 para 3 Gb (0 X c 0000000). Nos espaços usuários, endereços lineares e endereços lógicos são idênticos.

No 80386, endereços lineares vão de 0 Gb para 4 Gb. Um endereço linear indica uma posição particular de memória dentro deste espaço. Um endereço linear não é um endereço físico --- isto é um endereço virtual. Um endereço lógico consiste de um seletor e um offset. O seletor indica para um segmento e o offset diz que distância na seção o endereço é localizado.

O código Kernel e o segmento de dados são seções privilegiados definidos na tabela descritora global e extende de 3Gb para 4Gb. O Swapper - page - dir é organizado para que estes endereços lógicos e físicos sejam idênticos no espaço Kernel.

O espaço 3Gb acima aparece no process page directory como indicadores para tabelas de páginas Kernel. Este espaço é invisível para o processo no user mode, mas o modo privilegiado é acionado, por exemplo, para sustentar um sistema de ligação. O modo surpevisor é inserido dentro do contexto do processo atual então a tradução do endereço ocorre com respeito ao diretório de página do processo, mas usando segmentos Kernel. Isto é idêntico no mapeamento produzido com o uso de swapper - pg - dir e segmentos Kernel como ambos diretórios de páginas usa a

mesma tabela de página neste espaço. Apenas task [0] (A tarefa inativa, ás vezes chamada de "tarefa trocadora" por razões históricas, mesmo assim isto não tem relação com trocas nos implementos Linux) usa o swapper - pg - dir diretamente.

- O segmento base do processo usuário = o X 00, page - dir particular, para o processo.
- O processo usuário faz um sistema de ligação : segment base = 0 X c 0000000 page - dir = mesmo usuário page dir.
- swapper - pg - dir contém um mapeamento para todas as páginas físicas de 0 X 0000000 para 0 X c 0000000 + and_mem, então as primeiras 768 entradas em swapper - pg - dir são 0's, e então há 4 ou mais que indicam na tabela de páginas Kernel.
- O user page directories têm as mesmas entradas como swapper - pg - dir dos 768 acima. As primeiras 768 entradas mapeam o espaço usuário.

A vantagem é que sempre que o endereço linear é acima de 0 X c 0000000 tudo usa a mesma tabela de páginas Kernel (Kernel page Tables).

O monte usuário permanece no topo do segmento de dados do usuário e desce. O Kernel Stack não é uma bonita estrutura ou segmento de dados que eu possa apontar com um "aqui é um Kernel Stack". Um Kernel Stack_frame (uma página) é associada com cada novo processo criado e é usado sempre que o Kernel opera dentro do contexto deste processo. Coisas ruins aconteceriam se Kernel Stack descesse abaixo de seu corrente stack frame. [Onde o Kernel Stack é guardado? Eu sei que há um para cada processo, mas onde isto é armazenado quando isto não está sendo usado?]

Páginas usuários podem ser roubados ou trocados - Um user page é um que é mapeado abaixo de 3 Gb em uma tabela de páginas usuários. Esta região não contém page directories ou page tables. Apenas páginas sujas são trocadas.

Menores alterações são necessárias em alguns lugares (testes para limites de memória vem para a mente) para prover suporte para definidos segmentos programados. [Há agora uma modificação - |c| + O sistema de ligação usado por dosane, Wine, Twin, and Wabi para criar segmentos arbitrários.]

4.2 - Memória Física

Aqui está um mapa de memória física antes que qualquer processo de usuário for executado. A coluna da esquerda mostra o endereço de partida do item e os números em negrito são aproximados.

A coluna do meio mostra os nomes dos itens. A grande coluna da direita mostra a rotina relevante ou o nome variável ou explicações para ingresso.

- Projeto - Inits que adquirem memória são (principais.c) profil - buffer, com, init, psaux, init, rd, , init, scsi.dev - init.

Note que toda memória não marcada como livre é reservada (mem-init). Páginas reservadas pertencem ao Kernel e nunca estão livres ou trocadas.

Uma visão de memória do user process.

O código de segmento e dados do segmento extendem todo o caminho de 0 X 00 para 3 Gb. Correntemente o page fault handler do wp_page confere para assegurar que um processo não escreve para seu código de espaço.

De qualquer modo, pegando o sinal segu, é possível escrever para o code space, causando ocorrência de um copy - on - write. O Handler do_no_page assegura que qualquer página nova que o processo adquira pertença ao executável, uma biblioteca dividida, ao stack, ou dentro do valor do brK.

Um usuário de processo pode reordenar seu valor brK chamando sbrK ( ). Isto é o que malloc ( ) faz quando precisa. O texto e a porção de dados são distribuídos em páginas separadas ao menos que alguém escolha o N opção composta. A biblioteca dividida carrega endereços são correntemente tornadas da imagem dividida por ele mesmo.

O endereço é entre 1.5 Gb e 3 Gb, exceto em casos especiais.

4.3 - Distribuição da memória do processo usuário

O Stack, shlibs e os dados são muito afastados um do outro para serem spanned por uma tabela de página. Todas KPT são divididas por todos processo e deste modo eles não estão na lista. Apenas páginas sujas são trocadas. Páginas limpas são roubadas e deste modo o processo pode tê-los de volta para o executável se for desejado. A maioria das vezes apenas as páginas limpas são divididas. Uma página suja termina dividida sobre um fork até que parent ou child escolham para escrever isto de novo.

Administração dos dados da memória na tabela do processo.

Aqui está um sumário de algum dos dados mantidos na tabela do processo que é usado para administração da memória.

Limites do processo da memória

Ulong - start_code - and_code - and_data - brk, atart - stock

Erro de contagem de página

Tabela do descritor local
Sturct desc - sturct ldt {32} é a mesa descritora local para tarefa. Números de páginas residentes. Swappable - trocáveis Se então as páginas do processo não serão trocados. Kernel Stack page Indicador para a página distribuída no fork. Saved - Kernel - Stack V86 modo material (stuff) stract tss pilha de segmentos (stack segments) indicador da pilha Kernel Kernel stack pointer segmento da pilha Kernel Kernel stack segment (0X10) ssi = esp 2 = ss2 = 0

Níveis de previlégio não usados.

Segmentos seletores

DS=ES=FS=GS=SS=OK17,CS

Todos indicam para segmentos no corrente 1 dt [ ]
c r 3 : indicam para o page directory para este processo
1 dt - LDT (n) seletores para tarefas correntes do LDT

4.4 - Inicialização da memória

No Start Kernel (main.c) há 3 variáveis relatadas para inicialização da memória:

memory_start começa a 1 Mb atualizado pelo projeto de inicialização.
memory_end término da memória física: 8 Mb, 16 Mb, ou qualquer outro.
Low memory_start término do código Kernel e dados que é carregado inicialmente

Cada projeto init tipicamente torna memory_start e retorna um valor atualizado, se distribui espaços no memory_start (simplesmente pegando-a). Paging init ( ) inicializa a page-tables no { \ tt swapper - pg - dir} (começando a 0 X 0000000) para cobrir toda a memória física do memory_start para memory_end. Na verdade o primeiro 4 Mb é feito no startup_32

(heads).memory_start é incrementado se quaisquer nova page-tables são adicionados.

A primeira página é zerada para bloquear os indicadores das referências do alçapão nulo no Kernel.

No sched_init ( ) o 1 dt e tss descritores para tarefa [0] são postos no GDT, e carregado para dentro do TR e LDTR (a única vez que isto é feito explicitamente). Um trap gate (0X80) é ordenado para system-call.( ).

A bandeira tarefa aninhada é desligada na preparação para entrada do modo usuário: O cronômetro é ligado. O task-struct para task [0] aparece por inteiro em < Linux / sched.h >

mem_map é então construído por mem_init ( ) para refletir o corrente uso das páginas físicas. Este é o estado refletido no mapa da memória física da seção anterior. Então Dinux move para dentro do modo usuário com um iret após empurrar o corrente ss, esp, etc.

Claro que o segmento usuário para task [0] são mapeados bem sobre os segmentos Kernel e deste modo a execução continua exatamente onde isto termina.

Task [0]

pg_dir = swapper - pg - dir que sigmifica apenas endereços mapeados estão no alcance 3 Gb para 3 Gb + High memory.
LTD [1] = código usuário, base = 0 x 0000000, tamanho = 640 K
LDT [2] = dados usuários, base = 0 x 0000000, tamanho = 640 k

O primeiro exec ( ) põe a LTD entrada para task [1] para os valores usuários da base = 0x0, limite = task_size = 0 x c 0000000. Depois disso, nenhum processo vê os segmentos Kernel enquanto no modo usuário.

Processos e a Administração da Memória

Memória relacionada trabalho feito por fork ( ):

- distribuição de memória
- 1 página para o Task-struct
- 1 página para o Kernel Stack
- 1 para o pg_dir e algumas para pg_tables (cópias - páginas - tabelas)
- Outras mudanças
- sso põe para o segmento Kernel stack (0x10) para ter certeza?
- espo põe para o topo da nova distribuição Kernel - stack - page.
- c r 3 põe por copy - page - tables ( ) para indicar para nova página de diretório distribuída
- 1 dt = LDT (task_nr) cria novo 1 dt descritor
- descritores põe no gdt para novo tss e 1 dt [ ]
- Os registros restantes são herdados do parent.
- Os processos resultam dividindo seus códigos e segmentos de dados (embora eles tenham tabelas descritoras locais separados, as entradas indicam para os mesmos segmentos). O stack e páginas de dados serão copiados quando o parent ou child escreve para eles (copy-on-write).
- Memória relacionada trabalho feito por exec ( ):
- distribuição de memória
- 1 página para exec header para omagic
- 1 página ou mais para stack (max_arg_pages)
- clear-página-tables ( ) usado para remover páginas velhas.
- change 1 dt ( ) põe os descritores no novo 1 dt [ ]
- 1 dt [1] = código base = 0 x 00, limite = task - size
- 1 dt [2] = data base = 0 x 00, limite = task - size
- Estes segmentos são dpl = 3, p=1, s=1, g=1. Tipo = a (código or 2 dados)
- Eleva para MAX_ARG_PAGES páginas sujas de arqu e enup são distribuídos e guardado ao topo do segmento de dados para o novo usuário pilha criado.
- Ponha os indicadores de instrução do caller cip = ex.a_cutry
- Ponha o stack indicador do caller para o stack criado (esp=stack indicador). Este serão eliminados do Stack quando o caller resume.

- Limites de Memória Atualizados
- cud_code = ex.a_text
- cud_data = cud_code + &x.d_data
- brK = end_data + ex.ª_bss

Interrupções e traps são sustentadas dentro do contexto da corrente tarefa. Em particular, o diretório de páginas do corrente processo é usado na tradução de endereços. Os segmentos, de qualquer modo, são segmentos Kernel para que todos os endereços lineares apontem para dentro da memória Kernel quer acessar uma variável no endereço 0 x 01. O endereço linear é 0 x 00000001 (usando segmentos Kernel) e o endereço físico é 0 x 01. O último é porque a página do processo diretório mapea esta extensão exatamente como page_pg_dir.

O espaço Kernel (0 x c 0000000 + high - memory) e mapeado pela tabela de páginas Kernel que são eles mesmos parte da memória reservada. Eles são consequentemente divididas por todos processos. Durante um fork copy-page-tables ( ) trata tabela de páginas reservadas diferentemente. Isto põe indicadores no diretório de páginas de processo para indicar para tabelas de página Kernel e na verdade não distribui novas tabelas de páginas como isto faz normalmente. Como um exemplo o Kernel - Stack - page (que ocupa algum lugar no espaço Kernel) não precisa de um associado page - table distribuídos no pg-dir do processo para mapeá-lo.

O interruptor de instruções põe o indicador stack e o segmento stack do privilégio valor salvo no Tss do corrente task. Note que o Kernel stack é um objeto realmente fragmentado - Isto não é um objeto único, mas sim um grupo de stack frames. Cada um distribuído quando um processo é criado e deixado quando ele sai. O Kernel stack não deveria crescer tão rapidamente dentro de um contexto de um processo que extende abaixo da corrente frame.

4.5 - Adquirindo e liberando memórias

Quando qualquer rotina Kernel precisa de memória isto acaba chamando get-free-page ( ). Este está num nível mais baixo do que Kmallor ( ) (de fato Kmalloc ( ) get-free-page ( ) quando isto precisa mais memória).

Get-free-page ( ) toma um parâmetro, a prioridade.

Possíveis valores são gfp_buffer_gfp, Kernel, gfp,nfs e gfp atomic. Isto tira uma página do the free-page-list, atualizados mem_map, zeram a página e retorna o endereço físico da página (note que Kmalloc) retorna um endereço físico. A lógica do mm depende do mapa da identidade entre o endereço lógico e físico.

Isto é por ele mesmo bastante simples. O problema é claro, é que o free-page-list pode estar vazio. Se você não requisitar uma operação atômica, nesta etapa, você entra dentro do domínio de uma page stealing e que nós discutiremos em um momento. Como um último recurso (e para requisitos atômicos) uma página é separada do secundary-page-list (como você pode ter achado, quando páginas são libertadas, o secundary-page-list enche primeiro a manipulação atual da page-list e mem-map ocorre neste misterioso macro chamado remove-from-mem-queve ( ) que você provavelmente nunca quer investigar. O suficiente para dizer que interrupções são incapacitados. [Eu penso que isto deveria ser explicado aqui. Isto não é tão difícil...]

Agora de volta ao "Roubando páginas" get-free-page ( ) chame try-to-free-page ( ) que chame repetidamente shrink_buffers ( ) e swap-out ( ) nesta ordem até conseguir liberar uma página. A prioridade é aumentada em cada iteration sucessiva para que estas duas rotinas processem suas page-sterling-loops mais frequentemente. Aqui está um exemplo do processo swap-out:

- Faça a tabela do processo e adquira uma swappable task, por exemplo, Q.
- Ache um user page-table (não reservado) no espaço de Q.
- Para cada página na tabela try-to-swap-out (page)
- Termina quando a página é liberada.

Note que swap-out ( ) (chamada try-to-free-page ( )) mantém variáveis estatísticas e deste modo isto pode resumir a procura onde terminar a chamada

anterior try-to-swap-out ( ) examine os page-tables de todos usar process e obrigue o sterling policy:

1) Não brincar com as páginas (reserved) reservadas
2) Envelheça a página se ela é marcada acessada (1 bit)
3) Não mexa com página adquirida recentemente (last-free-pages ( ))
4) Deixe páginas sujas com map-counts > 1 intocadas
5) Diminua o map-count das páginas limpas
6) Librere páginas limpas se elas não são mapeadas
7) Troque páginas sujas com um map-count de 1

De todas essas ações, 6 e 7 vão parar o processo porque eles resultam na liberação atual de uma página física.

A 5ª ação resulta uma dos processos perdendo uma página limpa não dividida que não foi acessada recentemente (diminuindo Q à rss) que não é tão ruim, mas os efeitos cumulativos de algumas iterations pode atrasar o processo muito. No presente, há 6 iterations, deste modo uma página dividida por 6 processos pode ser roubada se está limpa. Page table então são atualizados e o TLB invalidado. O trabalho atual de liberar uma página é feito por free-page ( ), a complementação de get-free-page ( ). Isto ignora páginas reservadas, atualiza mem-map, e libera a página e atualiza o page-list (s) se não é mapeada. Para troca (em 6 em cima), write-swap-page ( ) é chamada e não faz nada notável da perspectiva da administração da memória. Os detalhes de shink-buffers ( ) nos levaria muito longe. Essencialmente isto procura free "buffers" (buffers são uma parte da memória que segura informação temporariamente quando dados transferem de um lugar para outro) em seguida escreve buffers sujos, e depois começa com buffers ocupados e chama free-page ( ) quando pode liberar todos os buffers numa página.

Note que page directories, page-table, e reserved pages não são trocadas, roubadas ou envelhecidas. Eles são mapeadas no process page directories com reserved page tables. Eles são liberados somente na saída do processo.

The page Fault Handles

Quando um processo é criado por fork, ele começa com um page directoru e uma página ou mais do executável. Deste modo the page fault handles é a forte da maioria da memória do processo. The page fault handles do page-fault ( ) recupera o endereço faltando no registro c r 2. O código do erro (recobrado no sys-call.s) diferencia o acesso do user / supervisior e a região para o fault-write proteção de uma página faltando. O anterior é sustentado pelo do-wp-page ( ) e o posterior pelo do-no-page ( ). Se o endereço falatando é maior do que Task-Size, o processo recebe um SIGKILL [Por que este controle? Isto pode acontecer somente em Kernel mode por causa da proteção do nível do segmento. Estas rotinas tem algumas sutilezas como elas podem ser chamadas num interrompimento. Você não ode supor que é a tarefa corrente que está executando de-no-page ( ) sustenta três situações possíveis:

1) A página é trocada
2) A página pertence a biblioteca executável ou dividida.
3) A página está faltando – uma página de dados não foi distribuída

Em todas as causas get-empty-pgtable ( ) é chamada primeiro para assegurar a existência de uma page table que cobre o endereço falatando. No terceiro para providenciar uma página no endereço requerido e no caso de uma página trocada, swap-in ( ) é chamado. No segundo caso, o handles calls share-page ( ) para ver se a página pode ser dividida com algum outro processo. Se isto falhar leia a página do executável ou biblioteca (Isto repete a chamada para Share-page ( ) se um outro processo fez o mesmo enquanto isso). Qualquer porção da página fora do valor brK é zerada.

A página lida do disco é contada como um erro maior. Isto acontece com um swap-in ( ) ou quando é lida da executável ou uma biblioteca. Outras casos são consideradas erros menores (mim-flt). Quando uma página divisível é achada ela é corite-protected. Um processo que escreve para uma página dividida vai precisar passar por um do-wp-page ( ) que faz o copy-on-write.

Do-wp-page ( ) faça o seguinte:

- Mande SIGSEGV se qualquer usar process o está escrevendo para o corrente code-space.
- Se a página velha não é dividida, então simplesmente não proteja-o. Senão get-free-page ( ) and copy-page ( ). A página adquirire a bandeira suja da página velha. Diminua a conta do mapa da página velha.

4.6 - Paginando (Paging)

Paginando é a troca numa base da página melhor do que os processos inteiros. Nós vamos usar trocando aqui para referir à "paginando" , uma vez que apenas Linux página, e não trocar, e pessoas são mais acostumadas à palavra "Swap" / "trocar" do que "page" / "paginar". Kernel pages nunca são trocadas páginas limpas também não são escritas para trocar. Elas são liberadas e recarregadas quando é requerida. O trocador mantém um único bit de informação de envelhecimento nas Páginas acessadas bit da page table cutries - [O que são os detalhes de manutenção? Como isto é usado?]

Linux suporta múltiplos swap files ou projetos que podem ser ligados ou desligados pelas ligações de swapoff system. Cada swap file ou projeto é descrito por uma strut-swap-info.

O campo das bandeiras (SWP-USED ou SWP-WRITE ok) é usado para controlar acesso para o swap files. Quando SWP- WRITE ok é desligado, o espaço não vai ser distribuído neste arquivo. Isto é usado por Swapoff quando isto tenta de não usar um arquivo. Quando swapoff adiciona um arquivo de troca nova isto aplica SWP-USED. Um variável imóvel no Swap files armazena o número dos arquivos ativos correntemente ativos. Os campos lowest - bit e hihgest - bit limitam a região livre na pasta de troca e são usadas para adiantar a procura por espaço de troca livre.

O programa do usuário m | < swap inicializa um swap device ou file. A primeira página contém uma assinatura (swap-space) nos últimos 10 bytes, e contém um mapa de bit. Inicialmente 1's no bitmap significam páginas ruins A'1' no bitmap significa que a página correspondente é livre. Esta página nunca é distribuída deste modo a inicialização precisa ser feita somente uma vez.

The Syscall Swapor ( ) é chamado pelo user program swapon tipicamente de / etc / rc. Algumas páginas da memória são distribuídas por swap-map e swap-lockmap, swap-map contém um byte para cada página no swapfile. Isto é inicializado do bitmap para conter 0 para páginas disponíveis e 128 para páginas que não pode ser usadas. Isto é para manter uma conta das petições da troca em cada página no swap file. Swap-lockmap contém um bit para cada página que é usada para assegurar exclusão mútua quando lendo ou escrevendo swap-files.

Quando uma página da memória está para ser trocada, um índice para posição da troca é obtido com uma chamada para get-swap-page ( ). Este índice é deste modo guardado em bits 1-31 da page table entry para que a página trocada possa ser localizada pela page fault handles, do-no-page ( ) quando necessário.

Os 7 bits mais altos do índice dão o swap file (ou projeto) e os 24 bits mais baixos dão o número da página neste projeto. Isto faz até 128 swap files, cada um com espaço para mais ou menos 64 Gb, mas o espaço em cima devido o swap map seria grande. Ao invés o tamanho do swap file é limitado para 16 Mb, porque o swap map então toma 1 página.

A função swap-duplicate ( ) é usado por copy-page-tables ( ) para deixar o processo da child herdar páginas trocadas durante um fork. Isto somente incrementa a conta mantendo no Swap-map para aquela página. Cada processo vai trocar numa cópia da página separa quando acessá-la. Swap-free diminui a conta mantendo no swap-map. Quando a conta abaixa para 0 a página pode ser redistribuída por get-swap-page ( ). Isto é chamado cada vez que uma página trocada é lida na memória (swap-inc) ou quando uma página está para ser descartada (free-one-table ( ), etc).

4.7 - Gerenciamento de Memória Cache

4.7.1 - Arquitetura de Memória Cache do Linux *(*Linux *Flush Architecture)*

O TBL é mais uma entidade virtual do que um modelo estrito quanto a Linux flush architecture e concernida. As característica única são isto mantem em ordem o mapeamento do processo kernel de algum modo, queira softivare ou hardware.

Código específico de arquitetura pode precisar ser modificado quando o kernel tiver mudado um processo/mapeamento kernel.

O shell (um lugar seguro p/ guardar dinheiro ou coisas) esta entidade é essencialmente "memory state"/"estado da memoria" como o flush architecture o vê. Em geral isto tem as propiedades seguintes:

- Isto sempre vai segurar cópias de dados que podem ser visto como atualizado pelo processo local.
- O funcionamento próprio pode ser relacionado ao TLB e o mapeamento do processo/Kernel page de algum jeito, isto é para dizer que eles podem depender um do outro.
- Isto pode, numa configuração cached virtual, causar problemas "aliasing" se uma página fisica é mapeada no mesmo tempo da que duas páginas virtuais e por causa dos bits de um endereço usado para catalogar a linha cache, a mesma porção do dedo pode acabar residindo no cache duas vezes, deixando resultados incompativéis.
- Projetos e DMA podem ou não ter capacidade para ver a cópia de um dedo mais atualizado que resida no cache do processo local.
- Corretamente, é suposto que a coerência num ambiente multiprocessador é mantida pelo subsistema cache/memória. Isto que dizer que, quando um processador requerer um dado no memory bus de maneira e um outro processador tem uma cópia mais atualizada, de qualquer jeito o requesitor vai obter uma cópia atualizada que pertença um outro processador.

NOTA: Arquiteturas SMP sem hardware, cache conferece mechanísms são realmente possíveis, o arquitetura current flush não sustenta isto corretamente, se em algum ponto o Zinux apontar em algum sistema onda isto é uma questão debatida, eu vou adicionar os ganchos necessários mas não vai ser bonito)

Sobre o que o Fluch Architecture se importa: sempre, a visão da administração de memória hardware de um conjunto de mapeamento do processo Kernel serão consistentes com aqueles do Kernel page tables.

Se o memory managemat kernel code faz uma modificação para a user process page modificando o dado via kernel space alias da página física subjacente, o fio controle de usuário vai ser o dado correto antes que é permitido continuar a execução, indiferente da cache architecture e/ou a semântica.

Em geral, quando o estado do espaço de endereço é mudado somente (em código genérico da administração da memória kernelnome de generic kernel management cade) o fluch architecture hook apropriado vai ser chamado descrevendo que o estado muda totalmente.

Sobre o que o flush architecture não importa: que o mapeamento do DMA "DMA/driver coerência. Isto inclui DMA mappings (no sentido do MMU mappings) e o cache/DMA dado consistência. Estes tipos des assuntos não devem esta no flush architecture, veja embaixo como eles devem ser manuseados.

Split Instrution/data cache consistência com respeitro as modificações feito para processo de instrução de espaço realizado pelo código de sinal de despacho signal dispatch cade. De novo, veja embaixo como isto devem ser manuseado de um outro jeito.

As interfaces para a flushachitesture e como executá-los em geral todas as rotinas descritos embaixo vão ser chamados na sequência seguinte: Fluch-cache-foo(...);

```
modify-address-space ();
clush - tlb-foo (...)
```

A lógica aqui é: Isto pode ser ilegal num arquitetura dada por um pedaço de dado cache para ensitir quando o mapeamento por aquele dado não existe, portanto o flush deve ocorrer antes que a mudança é feita.

É possível para uma arquitetura de MMU/TLB dada realizar um andamento da tabela hardware hardware table wolk dos kernel page tables, portanto o TLV flush é feito depois que os page tables terem sido mudados para que depois o hardware só pode carregar a cópia nova da informação de page table para o TLB

```
void flush - cache - all (void);
void flush - tlb - all (void);
```

Essas rotinas são para notificar o architecture specific code que mapeamento do espaço do endereço kernel uma mudança foi feita ao kernel address space mappings, que significa que os mapeamentos de todos processos foram efetivamente mudados.

4.7.2 - Implementação da Memória Cache

Uma implementação deve:

- Eliminar todos os entradas do cache que são válidas neste momento quando flush-cache-all é invocado isto refere-se ao virtual cache architecture, se a cache is write-back, essa rotina vai submeter o dado da cache para memoria antes do que invalidar cada ingresso. Para caches físicos, não é necessário realizar uma ação já que mapeamento físico não tem ponto de apoio no address space translations.

- Para flush-tlb-all todos TLB mappings para o kernel address space devem ser feito consitente com os OS page tables de qualquer maneira. Norte que com um arquitetura que possua a nação

- Para flush-tlb-mm, o tlb/mmu hardware é para estar localizado num estado onde isto vai ver (agora corrente) kernal page table entradas para o espaço de endereço pelo mm-strust.

```
flush_cache_range(struct mm_struct *mm, unsigned long start,
                  unsigned long end);
flush_tlb_range(struct mm_struct *mm, unsigned long start,
                unsigned long end);
```

Uma chance para uma particular range do user address no adelrass space descrito pelo mm-struct passada esta ocorrendo. As duas notas acima para FLUSH - mm( ) relecianando a mm-struct passada aplicam-se aqui também.

- Para Flush-cache-range num virtualmente cached system, todas entradess cache que são nolidas pena a range partem para o fim no address space descrito pelo mm-struect são para ser invalidadas.

- Para Flush-tlb-range, qualquer ação necessária para causar o MMUITLB hardware não conter traduções estragados são para ser realizados. Isso significa que quaiquer traduções estão no Kernel page tables no range start para acabar no address space descrito pelo mm-struet são para que a administração da memoria hardware sera deste ponto avançado, por qualquer significado.

```
void flush_cache_page(struct vm_area_struct *vma, unsigned long address);
void flush_tlb_page(struct vm_area_struct *vma, unsigned long address);
```

Uma chance para uma única página no address dentro do usar space para o address space descrito pelo um area-struet passado esta ocorrendo. Uma

efetivação, se necessária, pode obter na mm-struet associado para este address space via uma um - Flags. Este caminho em uma efetivação onde a instrução e dara space não são unificados, alguem pode conferir para ver se um-exee esta posto no uma-sum-flags para possivelmente avistar flushing o instruction space, por exemplos:

As duas notas acima para flush-*-mm( ) concermindo o mm-struct (passado indiretamente via uma -um-mm) aplica aqui também.

A implemetação deve também :

- Para flush-cache-range, num virtualmente cache systam, todas entradas cacha que são validas para a página no addrees no address space descrito pelo uma são para ser invalidados.

- Para flush-tlb-range, qualquer ação necessária para causar o MMU/TLB hardware para não conter traduções estragadas são para ser efetuadas. Isto significa que quaisquer traduções estão nos kernel page tables para a página no address space descrito pelo uma passado são para que a administração de memória hardware, serão vistas deste ponto avançado de qualquer maneira.

4.7.3 - Carregando o Flush-PAGE para a RAM (Unsigned Long Page);

Este é o patinho feio. Mas sera semântica é necessário em muitas arquiteturas que precisei para adicionar isto para a arquitetura flush para Linux. Brevemente, quando (como um exemplo) serve um kernel um enode cow, isto usa o "suposto" mapeamento de todas memórias físicas no espaço kernal para efetuar a cópia da página em questão para uma nova página. Este apresenta um problema para caches virtualmente catalogados que são write-back escritos de volta na natureza. Neste caso, o Kernel toca duas páginas físicas no espaço Kernel. A seqüência do código sendo descrito aqui essencialmente parece como:

```
do_wp_page()
{
 [ ... ]
        copy_cow_page(old_page,new_page);
        flush_page_to_ram(old_page);
        flush_page_to_ram(new_page);
        flush_cache_page(vma, address);
        modify_address_space();
        free_page(old_page);
        flush_tlb_page(vma, address);
 [ ... ]
}
```

Alguns dos códigos atuais tem sido simplificados para propósitos específicos.

Considere um cache virtualmente catalogados que é escrito de volta write-back. Neste momento que a cópia da página acontece para o suposto espaço kernel, é possível para usar space a visão da página original para estar no caches (no endereço do usuário, por exemplo, onde o erro esta ocorrendo). A cópia da página pode trazer este dado (para a página velha) dentro do caches. Será também colocado o dado (no novo suporte kernel mapeado da página) sendo copiado para dentro da cache, e para write-back escrever de volta cache este dado vai ser sujo ou modificado no cache.

Em tal caso a memória principal não será a cópia mais recente do dado. Os caches são estúpidos, então para a nova página que estamos dando ao usuário, sem forçar o dado cached no suposto kernel para a memória principal o processo será o conteúdo velho da página. (Por exemplo qualquer lixo que estarem lá antes da cópia ter sido feita pelo processamento COW acima).

4.7.3.1 - Exemplo concreto de flush-page

Considere um processo que divide uma página, lê somente READ-ONLY com maior uma tarefa (ou varias) no endereço virtual Ox2000, no usar space. E para propósito específicos deixe nos dizer que este endereço virtual mapeia para a página física 0x14000.

Se a tarefa 2 tenha escrever para a página lê apenas no endereço 0x2000 nós alteremos um esso e (eventual fragmento do código) mente resultado no code fragment mostrando acima no do-WP-PAGE ( ).

O Kernel vai obter uma nova página para tarefa 2, deixe-nos dizer que esta e uma página física 0x2600, e deixe-nos também dizer que os mapeamentos do suposto Kernel para páginas físicas 0x14000 e 0x26000 podem residir em dias únicos linhas cache ao mesmo tempo buscando no esquema da linha catalogada deste cache.

O conteúdo da página e copiado do mapeamento Kernel para página física 0x14000 para uns para página física 0x26000.

Neste momento, numa arquitetura cache virtualmente catalogada write - back nos temos uma inconsistência potencial. O novo dado copiado dentro da página física 0x26000 não e necessário na memória principal neste momento, de fato isto poderá estar toda no cache apenas no suposto kernel do endereço físico.

Também, o (não modificando, por exemplo, limpo) dado para a (velha) página original esta no cache do suposto kernel para página física 0x14000, isto pode produzir uma inconsistência mais tarde, então para proteger isto e melhor eliminar as cópias cached deste dado também.

Deixe-nos dizer não escrevemos os dados de volta para a página no 0x256000 e nos apenas deixamos isto lá. Nos retornaríamos para a tarefa 2 (Quem teve esta nova página agora mapeada no endereço virtual 0x2000) ele completaria sua escrita, então ele leria algumas outras porções de dados nesta nova página (por exemplo, esperando o conteúdo que existe lá antes). Neste momento seu dado é deixado no cache no suposto kernel para nova página física, o usuário obterá o que estava na memória principal antes da cópia para sua leitura. Isto pode levar a resultados desastrosos.

4.7.4 - Conteúdo de uma arquitetura virtual

Numa arquitetura cache virtualmente catalogada, fica o que foi necessário para fazer a memória principal consistente com a cópia cached da página passada do espaço kernel.

Nota: Isto é na verdade necessário para esta rotina invalidar linhos em um cache virtual que não escrito de volta é write - back na natureza. Para ver porque isto e realmente necessário, refaça o exemplo acima com a tarefa 1 e 2, mas agora fork() ainda outra tarefa 3 antes dos erros do cow ocorreram, considere o conteúdo do caches no kernel e user space se a seqüência seguinte ocorre na exata sucessão:

1. Tarefa 1 lê uma parte da página no 0x2000
2. Tarefa 2 COW erra a página no 0x2000
3. Tarefa 2 efetiva suas escritas para a nova página no 0x2000
4. Tarefa 3 COW erra a página 0x2000

Mesmo em um cache não escrito de volta virtualmente catalogado, a tarefa 3 pode ver o dado inconsistente depois do erro COW se FLUSH-PAGE-TO-RAM não invalida a página física do suposto kernel do cache.

VOID-UP-DATE

Embora não estritamente parte da arquitetura flush, em certas arquiteturas algumas operações e controles precisam ser efetuados aqui para as coisas darem certo proporcionalmente e para o sistema manter-se consistente.

Em particular, para caches virtualmente catalogados esta rotina deve conferir para ver que o novo mapeamento que vem sendo adicionado pelo corrente erro de página não adiciona um bad alias "para o user space".

Um "Bad Alias" é definido como dois ou mais mapeamentos (pelo menos um dos quais é possível de ser escrito) para duas ou mais o páginas que traduzem para a exata página física, e devido ao algarismo catalogado do cache pode também residir na única e mutuamente exclusiva linhas cache.

Se um BAD ALIAS é detectado, uma implementação precisa resolver esta inconsistência de alguma maneira, uma solução e andar através de todo os mapeamentos e mudar as page-tables para fazer estas páginas como não concretizáveis se o hardware permite tal coisa.

As conferências para isto são muito simples, tudo que uma implementação precisa fazer é:

Se ((uma -Um - Flags 6 (Um - Write/Um - Shared)) confere sua potência mau supostas, então para o caso comum (mapeamento escritos devidos são extremamente raros) apenas uma comparação é necessitada para sistemas COW CACHES virtualmente catalogados.

4.7.5 - Implicações Referentes a Arquitetura

4.7.5.1 - Arquitetura baseada no Modelo SMP

Dependendo da arquitetura certos consertos podem ser necessários para permitir a arquitetura FLUSH para trabalhar num sistema SMP.

O principal assunto e se uma das operações FLUSH acima fazem que o sistema inteiro veja o FLUSH globalmente, ou o FLUSH e apenas garantido para ser visto pelo processador local.

Em um último caso um CROSS CALLING MECHANISM é necessário. Os dois correntes sistemas SMP suportados no Linux (intel e space) usam inter-processor interrupts para "transmitir" a operação FLUSH e faz isto correr localmente em todo processador se necessário como um exemplo, no sistema SUNHM Space todos processadores no sistema precisam executar o pedido FLUSH para garantir a consistência através do sistema inteiro. De qualquer modo, nas máquinas SUNHD Space, TLB FLUSHES efetivamente no processador local são transmitidos sobre o BUS-SYSTEM pelo hardware e desta forma uma ligação cruzada não e necessária

4.7.5.2 - Implicações para arquitetura baseados no contexto MMU/CACHE.

A idéia inteira por trás do conceito de MMU e facilidades do contexto cache é para permitir muitos ADDRESS SPACES para dividir os recursos CACHE/MMU no CPU.

Para levar total vantagem de tal facilidade, e ainda manter a coerência descrita acima, requer-se algumas considerações extras do implementador.

As questões envolvidas variam muito de uma implementação para outro, pelo menos esta tem sido a experiência do autor. Mas em particular algumas destas questões são provavelmente para ser:

- A relação do mapeamento do espaço Kernel para os USER-SPACE, num contexto são **convertidas**, alguns mapeamentos do sistema kernel tem um atributo global, naquele o hardware não **concerde** ele mesmo com o contexto da informação quando uma tradução é feita, que tem seu atributo. Desta forma um FLUSH (em qualquer contexto) de um mapeamento de um Kernel CACHE/MMU poderia ser suficiente.

De qualquer maneira e possível um outros implementações para o Kernel para dividir o contexto chave associado com um ADDRESS SPACE particular. Pode ser necessário em tal caso andar por todos contextos que são correntemente válidos e efetuam o Flush completo em cada um para um Kernall Address Space Flush.

O custo por contexto Flush podem tornar uma questão chave, especialmente com respeito ao TLB. Por exemplo, se um Tlb Flush e necessário, em um grande Range de endereços (ou um inteiro Address Space) pode ser mais prudente distribuir e assumir um nova contexto MMU/para este processo por causa da eficiência

4.7.6 - Como tratar o que a arquitetura flush não executa com exemplos

A arquitetura Flush descrita não faz emendas para coerência de projetos DMA com dados Cached. Isto também não tem provisões para nenhuma estratégia de

mapeamento necessários pelo DMA e projetos se forem necessários em um certa máquina Linux é Portad To. Nenhuma destas questões são para a arquitetura Flush.

Tais questões são negociadas mais claramente no nível do Driver do projeto. O autor está mais convencido disto depois de sua experiência com um conjunto comum de sparc device drivers que precisaram de toda função corretamente em mais do que uma hand full de cache/mmu e bus architetures no mesmo kernel. De fato esta implementação é mais eficiente porque o motorista sabe exatamente quando o DMA precisa ver o dado consistente ou quando o DMA está indo criar uma inconsistência que deve ser resolvida. Nenhuma tentativa para atingir este nível de eficiência via colchetes soma ao código de administração genérica da memória kernel seria complexo e muito obscura como um exemplo, considere no sparc como os DMA buffers são manuscrito. Quando um device driver deve efetuar o DMA para/de um único buffer, ou uma dispersa lista de muitos buffers, ele usa um conjunto de rotinas abstratas.

```
Char * (*mmu_get_scsi_one)(char de char *, unsigned Linux_sbus longo de struct *sbus);
 sem  (*mmu_sglist (*mmu_get_scsi_sgl)(struct de efeito *, int, Linux_sbus de struct *sbus);
 sem  (*mmu_release_scsi_one)(char de efeito *, unsigned Linux_sbus longo de struct *sbus);
 sem (*mmu_sglist (*mmu_release_scsi_sgl)(struct de efeito *, int, Linux_sbus de struct *sbus);
 sem  (*mmu_map_dma_area)(unsigned de efeito addr longo, len de int);
```

Essencialmente o mmu_get_* rotinas são passadas por um indicador ou um conjunto de indicadores e especificações de tamanho para áreas no espaço kernel para que o DMA ocorra, eles retornam para o endereço capaz do DMA (por exemplo um que pode ser carregado do controlador do DMA para o transferidor). Quando o driver é feito como DMA e o transferidor tiver completado com o(s) endereço(s) DMA para que recursos possam ser liberados (se necessário) e cache flushes possam ser efetivados (se necessário). A rotina ter um bloqueio de memória de DMA por um longo período de tempo, por exemplo, um motorista de networking usaria isto para uma transmissão de pesquisa ou receber buffers. O argumento final é uma entidade especifica Sparc que permite o código do nível da máquina efetuar o mapeamento se o mapeamento do DMA são ordenados em uma base por-bus.

4.7.7 - Questões abertas na Arquitetura Cache

Há pareceres para muitas arquiteturas cache estúpidas lá fora que queiram causar problemas quando um Alias está situado dentro do cache (mesmo um protegido onde nenhuma das entradas do cache suposto são escreviveis!). Da nota está o mipsr4000 que dará uma exceção quando tal situação ocorre, elas podem ocorrer quando o processamento cow está acontecendo na corrente implementação. No mais chips que fazem algo tolo como isto, um exception handler pode flush as entradas no cache que está sendo reclamado e tudo está em ordem. O autor esta mais concernido sobre o custo dessas exceções durante o processamento cow e seus efeitos que ocorrerão na performance cow, que essencialmente está para flush um user space page e se não o fazendo então causaria os problemas acima descritos.

Tem sido tardiamente aquecida a conversa sobre muito inteligentes networking hardware. Pode ser necessário estender a arquitetura flush para prover as interfaces e facilidades necessárias para estas mudanças para o código networking. É claro que, a arquitetura flush é sempre sujeita a aperfeiçoamentos e mudanças para buscar novas questões ou novos hardwares que apresentam um problema que estava até este ponto desconhecido.

5. Sistema de Arquivo no Linux (File System)

5.1. - Conceitos Fundamentais

5.1.1 - Arquivos

Conceitualmente, arquivos são mecanismos de abstração que fornece uma forma de armazenar e recuperar informações em disco. A características mais importante de qualquer mecanismo abstração é a forma de identificar os objetos como os quais o mecanismo trata. Quando um processo cria um arquivo, é preciso que tal arquivo receba um nome, normalmente dado pelo processo. Quando tal processo

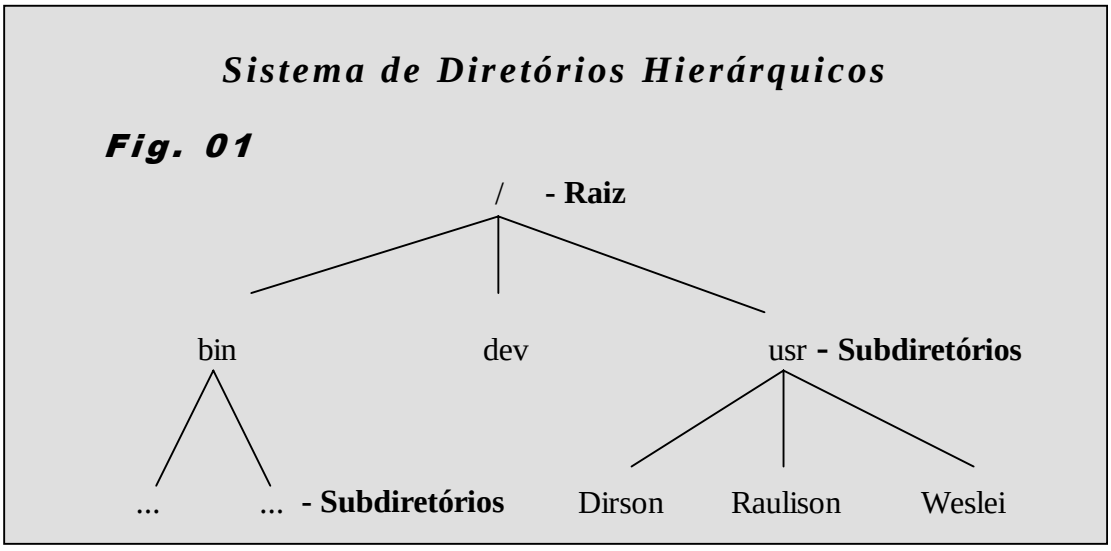
termina sua execução, o arquivo continua a existir, podendo ser acessado por outros processos, usando para tanto o nome atribuído ao arquivo.

O Linux faz distinção entre nome maiúsculos e minúsculos. Normalmente um nome de arquivo é composto de nome e uma extensão, separada por ponto no Linux, o tamanho da extensão, se houver, fica a critério do usuário, e uma arquivo pode até ter duas ou mais extensões, exemplo: prog.c.Z.

Não há limite de números de caracteres utilizados para dar nome a arquivos. O Sistema Operacional Linux, olha o arquivo como uma seqüência de byte, sem nenhuma estrutura, isto dá uma flexibilidade espantosa ao sistema de arquivo. Os programas de usuários, podem colocar o que desejarem nos arquivos e identificá-los da forma que lhe for mais conveniente, o Unix não influência em NADA nesta processo de identificação.

5.1.2 - Diretórios

Para tratar dos arquivos, o sistema operacional normalmente lança mão do diretórios, no caso do Linux diretórios hierárquico, vide figura 01. Os diretórios são um tipo de arquivo.



No Linux todos os arquivos fazem parte de um diretório, assim eles são mantidos e organizados, os diretórios são meios de oferecer endereços dos arquivos, de maneira que o SO possa acessá-los rapidamente e facilmente, ao entra pela primeira vez em sua conta, o usuário já esta em um subdiretório denominado subdiretório de entrada.

5.1.3 - Conta

É uma senha que é aberta pelo administrador do sistema (denominado de **root**) onde o usuário identifica-se para o computador, que então dá acesso ao seu diretório de entrada, onde você pode executar os comandos permitidos a sua senha. Nos SO padrão Unix, a conta é obrigatória para todos, a figura 02 mostra um exemplo de abertura de conta no Linux.

```
Linux 2.0.0 (carvalho.cpd.ufg.br) (tte p0)


carvalho login: root
Password:
Ast login: Wed Jan 29 12:16:37 from jacaranda.cpd.uf
Linux 2.0.0.
carvalho:~$
```

Figura 02

5.1.4 - Tipos de arquivos

O Linux suporta arquivos regulares, arquivos de diretório, arquivos especiais de caracteres e arquivos especiais blocados.

Os arquivos regulares são aqueles que contém informações de usuários, por exemplos, tipo ASCII. Arquivos diretórios são arquivos usado na manutenção do sistema de arquivo. Arquivos especiais de caracteres estão diretamente ligado à entrada/saída e são usados para dispositivos seriais de entrada/saída, tais como terminais, impressoras e rede. Os arquivos especiais blocados são usado modelar dispositivos. Um exemplo de tipos de arquivos utilizados no Linux pode ser visto na figura 03.

```
carvalho:/usr$ ls
X11@            etc/            lib/            spool@
X11R6/          games/          local/          src/
X386@           i486-Linux/     man/            tclX/
adm@            i486-Linuxaout/ opemwin/        tkX/
bin/            i486-sesv4/     préerve@        tmp@
dict/           include/        sbin/
doc/            info/           share/
ftpusers        mtools.conf     sesog.conf
carvalho:/usr$
```

Figura 03

5.1.5 - Acesso a arquivos

O SO Linux, bem como os demais SO, trata o acesso a arquivos de forma randômica, ou seja, seus byte ou registros podem ser lidos em qualquer ordem.

5.1.6 - Atributos dos arquivos

Cada arquivo tem necessariamente um nome e um conjunto dados. Além disso, o Sistema Operacional associa a cada arquivo algumas outras informações que chamaremos de atributos de arquivos. A figura 04, nos mostra alguns dos atributos dos arquivos.

```
carvalho:/etc$ ls -l
total 11
lrwxrwxrwx   1 root    root           9       Dec    9  14:01 rmt -> /sbin/rmt*
-rw-r--r--   1 root    root         743       Jul   31  1994 rpc
-rw-r--r--   1 root    root          86       Jan   28  1994 securette
-rw-r--r--   1 root    root       21394       Dec    9 14:22 sendmail.000
-rw-r--r--   1 root    root       23580       Jan    6 12:28 sendmail.cf
drwxr-xr-x   2 root    root        1024       Dec    9 13:59 skel/
-rw-r--r--   1 root    root         314       Jan    9  1995 slip.hosts
-rw-r--r--   1 root    root         342       Jan    9  1995 slip.login
lrwxrwxrwx   1 root    root          13       Dec    9 13:59 utmp -> /var/og/utmp
lrwxrwxrwx   1 root    root          13       Dec    9 13:59 wtmp -> /var/og/wtmp
-rw-r--r--   1 root    root          76       Mae    8  1995 e p.conf.example
```

Figura 04

Como vimos neste exemplo, o Sistema de Arquivo do Linux permite restringir o acesso aos arquivos e diretórios permitindo que somente determinados usuários possam acessá-los. A cada arquivo e diretório é associado um conjunto de permissões. Essas permissões determinam quais usuários podem ler, escrever, ou alterar um arquivo, e no caso de arquivos executáveis como programas, quais usuários podem executá-lo. Se um usuário tem permissão de execução de um diretório, significa que ele pode realizar buscas dentro daquele diretório, e não executá-lo como se fosse programa. Passaremos a explicar a codificação, escolhemos aleatoriamente o sétimo arquivo **skel/** da figura 04:

d	r	w	x	r	-	x	r	-	x	nome do arquivo
1	2	3	4	5	6	7	8	9	10	skel/

obs : o que está em negrito,caixa maior, corresponde a posição do arquivo skel/

- 1 - informa o tipo de arquivo (**d**⇨diretório,l ⇨ link, - ⇨demais arquivo)
- 2 - Permissões do Proprietário (**r** ⇨ leitura, , - não permitida leitura)
- 3 - Permissões do Proprietário (**w** ⇨ escrita, - não permitida escrita)
- 4 - Permissões do Proprietário (**x** ⇨ execução, - não permitida execução)
- 5 - Permissões do Grupo (**r** ⇨leitura, , - não permitida leitura)
- 6 - Permissões do Grupo (w ⇨ escrita, **-** não permitida escrita)
- 7 - Permissões do Grupo (**x** ⇨ execução, - não permitida execução)
- 8 - Permissões do Sistema (**r** ⇨ leitura, , - não permitida leitura)
- 9 - Permissões do Sistema (w ⇨ escrita, **-** não permitida escrita)
- 10 -Permissões do sistema (**x** ⇨ execução, - não permitida execução)

5.2 - Operações sobre arquivos

Os arquivos existem para armazenar informações e permitir a sua recuperação. As Chamadas de Sistemas mais comum relacionadas ao Sistema de Arquivo Linux são chamadas que operam sobre arquivos individuais ou envolvendo diretórios e sistema de arquivos como um todo .

A chamada **CREAT** não só cria um arquivo, mas também abre esta arquivo para escrita, indepedente do modo de proteção especificado para ele. O descritor de arquivo que a chama retorna, FDh, pode ser usado para escrever no arquivo. Se a chamada CREAT for executada sobre um arquivo existente, esta arquivo será truncado para o comprimento 0, desde que os direitos do arquivos assim o permitam.

Para que um arquivo existente possa ser lido ou escrito, é necessário que ele seja primeiramente aberto e se ele esta aberto para leitura, escrita ou para ambas as operações. Várias opções podem ser especificadas. O descritor de arquivo que a chamada retorna pode então ser usado para leitura ou escrita. Posteriormente, o arquivo deve ser fechado através da chamada **CLOSE**, cuja execução torna o descritor de arquivo disponível para ser novamente utilizado numa chamada CREAT ou **OPEN** subseqüente.

A chamada **READ** é utilizada para ler o arquivo, os bytes lidos vêm em posição corrente de leitura. O processo que faz a chamada deve indicar a quantidade de informação a ser lida e providenciar um buffer para possibilitar a leitura.

A chamada **WRITE**, os dados são escritos no arquivo, geralmente a partir da posição corrente. Se tal posição for a de final de arquivo, o tamanho do mesmo cresce. Se a posição corrente no momento da escrita estiver no meio do arquivo, os dados existente nesta posição estavam perdidos para sempre, pois a operação de write escreve os novos dados em cima dos antigos.

Apesar da maioria dos programas ler e escrever arquivos seqüencialmente, em algumas aplicações os programas devem ser capaz de acessar randomicamente qualquer parte do arquivo. Associado a cada arquivo, existe um ponteiro que indica a posição corrente do arquivo. Quando a leitura ou escrita for seqüencial, em geral, ele aponta para o próximo byte a ser lido ou a ser escrito. A chamada **LSEEK** têm três parâmetros: o primeiro do descritor de área para o arquivo, o segundo é a posição do arquivo, o terceiro informa se a posição é relativa ao inicio do arquivo, à posição corrente ou final do arquivo. O valor que o **LSEEK** retorna é a posição absoluta no arquivo após a mudança no ponteiro.

Para cada arquivo o Linux mantem o modo do arquivo (regular, diretório ou arquivo especial), seu tamanho, o instante da última modificação, e outra informações pertinentes. Os programas podem verificar estas informações, usando a chamada **STAT**. Seu primeiro parâmetro é o nome do arquivo. O segundo é um ponteiro para a estrutura onde a informação solicitada deve ser colocada.

As chamadas do sistema relacionadas com os diretórios ou com o sistema de arquivo como um todo, em vez de um arquivo específicos. Os diretórios são

criados utilizando as chamadas **MKDIR** e **RMDIR**, respectivamente. um diretórios o pode ser removido se estiver vazio.

A ligação de um arquivo cria uma nova entrada no diretório que aponta para um arquivo existente. A chamada **LINK** cria esta ligação. Os seus parâmetros especificam os nome originais e novo, respectivamente. As entrada do diretórios são removidas via **UNLINK**. Quando a última ligação para um arquivo é removida, é automaticamente apagada . Para um arquivo que nunca foi ligado, o primeiro **UNLINK** faz com que ele desapareça.

Os diretórios de trabalho é especificado pela chamada **CHDIR**. Sua execução faz com que haja mudança na interpretação dos nome dos caminhos relativos.

A chamada **CHMODE** torne possível a mudança do modo um arquivo, ou seja, de seus bits de proteção.

5.3 - Arquivos Compartilhados

Quando vários usuários estão trabalhando juntos em um projeto, ele comumente precisam compartilhar arquivos. Em decorrência disto, muitas vezes é conveniente que um arquivo compartilhado apareça simultaneamente em diretórios diferentes que pertençam a diferentes usuários. A conexão entre um diretório e um arquivo compartilhado é chamada de ligação (link). O próprio sistema de arquivo é um gráfico acíclico dirigido, ou **DAG**, em vez de árvore. No Linux os blocos do disco não são listados no diretório, mas numa estrutura de dados associada ao próprio arquivo. Esta estrutura é chamada nó-i, é a forma como o Linux implementa compartilhamento arquivo.

5.4 - Estrutura do Sistema de arquivos do LINUX Release 1.2

5.4.1 - Apresentação

Este trabalho é baseado na versão 1.2 da Estrutura do Sistema de arquivos do LINUX (LINUX **File System Structure**) **FSSTND,** que por sua vez é baseado em um documento de consenso da comunidade Linux (que poderá ser encontrado na internet -**www.**Linux **.org**) o layout do sistema de arquivos foi inicialmente desenvolvido dentro da lista de e-mail FSSTND do LINUX**-ACTIVISTS**.

O coordenador do FSSTND é Daniel Quinlan **<Daniel.Quinlan@**Linux**.org>.** Uma parte considerável deste trabalho foi tirado da **FAQ** (Lista de perguntas mais frequentes) mantida por Ian McCoghrie (**FSSTND-FAQ**). Este documento está disponível via ftp anonymous em tsx-11.mit.edu no diretório /pub/Linux/docs/Linux-standards/fsstnd/ FSSTND-FAQ

Nosso trabalho enfocará a estrutura do sistema de arquivos para LINUX típico, incluindo a localização de arquivos e diretórios, e o conteúdo de alguns arquivos de sistema.

5.4.2 - Características Sistema de Arquivos

O sistema de arquivos Linux esta caracterizado por:

- Uma estrutura hierárquica.
- Um tratamento consistente da informação dos arquivos.
- Proteção dos arquivos.

O sistema de arquivos Linux segue o mesmo princípio básico que a maioria dos sistemas de arquivos UNIX seguem. Apesar que o sistema de arquivo não concordar em 100% com cada aspecto possível de alguma implementação particular do sistema UNIX. De qualquer forma, muitos dos aspectos da implementação do sistema de arquivos estão baseados em idéias encontradas em sistemas similar ao UNIX system V, outros fatores também foram levado em conta tais como:

- Práticas comuns na comunidade LINUX.
- A implementação de outras estruturas de sistemas de arquivos.
- Definição das categorização ortogonal de arquivos: Compatível versus não compatível. e variável versus estáticos.

A informação compatível é aquela que pode ser compartida entre várias máquinas diferentes. A não compatível é aquela que deve ser localizada em uma máquina particular. Por exemplo, os diretórios local dos usuários são compatível, porém os arquivos de bloqueio do dispositivo (lock file) não são compatíveis.

A informação estática inclui arquivos, bibliotecas, documentação e tudo aquilo que não precisa da intervenção do administrador do sistema. A informação variável é tudo aquilo que se troca com a intervenção do administrador.

O entendimento destes princípios básicos ajudará a guiar a estrutura de qualquer sistema de arquivos bem planejado.

A distinção entre informação compatível e não compatível é necessária por várias razões:

- Em um ambiente de rede, existe uma boa quantidade de informação que se pode compartilhar entre diferentes máquinas para o aproveitamento de espaço e facilitar a tarefa da administração.

- Em um ambiente de rede, certos arquivos contém informação especifica de uma só máquina, portanto, estes sistemas de arquivos não podem ser compartilhados (antes de tomar medidas especiais).

- As implementações de fato do sistema de arquivos não nos permitem que a hierárquia /usr fosse montada somente para leitura, porque possuía arquivos e diretórios que necessitavam ser escritos muito freqüentemente. Este é um fator que deve ser atacado quando algumas parte de /usr são compartilhadas na rede.

A distinção "compatível" pode ser usada para suportar, por exemplo:

- uma partição /usr (o componente de /usr) montada (somente para leitura) através da rede (usando NFS).

- uma partição /usr (o componente de /usr) montada somente para leitura de um cd-rom, pode ser considerado como um sistema de arquivos somente para leitura, compartilhado com outros sistemas LINUX utilizando o sistema de e-mail como uma rede.

- A distinção "estática" contra "variável" afeta o sistema de arquivos de duas maneiras principais:
  - Arquivo da /(raiz) contém ambos tipos de informação, variável e estática necessita permitir leitura e escrita.
  - Arquivo do /usr tradicional contém ambos tipos de informação variável e estática e os arquivos poderíamos desejar montar-los somente para leitura, é necessário proporcionar um método para fazer que /usr funcione somente para leitura. Isto se faz com a criação de uma hierárquia /var que funciona leitura e escrita (ou é parte de uma partição leitura-escrita tal como /), que diminui muito a funcionalidade tradicional da partição /usr.

O diretório raiz/. Esta sessão descreve a estrutura do diretório raiz. O conteúdo do sistema de arquivos raiz será adequado para carregar, dar boot, restaurar, recuperar o sistema:

Para carregar o sistema, deve estar presente o suficiente como para montar /usr e outras parte não essenciais do sistema de arquivos.

Esta inclui ferramentas, informação do configuração e do boot de inicialização (boot loader) e algum outra informação essenciais ao inicializar.

Para habilitar a recuperação e/ou a reparação do sistema, estará presente no sistema de arquivos raiz aquelas ferramentas que o administrador experimentado necessitaria para diagnosticar e reconstruir um sistema danificado.

Para restaurar um sistema, estão presente no sistema de arquivos raiz aquelas ferramentas necessárias para restaurar o sistema em floppy, fitas, etc.

A principal preocupação e que procuraremos colocar muito pouca coisas do sistema de arquivos raiz, é a meta de manter tão pequeno quanto possível. Por várias razões é desejável mantê-lo pequeno.

É freqüentemente montado em mídia muitas pequenas. Por exemplo muitos usuários de LINUX instalam e recuperam sistemas montando como o disco copiado de disquete 1.44Mb.

O sistema de arquivos tem muitos arquivos de configuração específicos de um sistema. Possíveis exemplos são de um kern, que é específicos do sistema, um hostname diferente, etc. Isto significa que o sistema de arquivos / não é sempre compatível entre sistemas em rede. Mantendo-o pequeno no sistemas de rede, se minimiza o espaço perdido no servidor utilizados pelos arquivos não compatível. Também permite estações de trabalho com winchesters locais menores.

Os erros do disco, que corrompe as informação no sistema de arquivos / são um problema maior que os erros em qualquer outra partição. Um sistema de arquivos pequeno é menos propenso a perder arquivos como resultado de uma falha do sistema.

É recomendável colocar, principalmente os arquivos, em /etc/mtab. De qualquer forma, não se necessita que o sistema de arquivos está totalmente armazenado localmente. A partição / não tem porque estar armazenada localmente pode ser especificada pelo sistema por exemplo, poderiam estar montada de um servidor NFS.

O software não deverá criar ou pedir arquivos de subdiretórios especiais. A estrutura do sistema de arquivos LINUX proporciona suficiente flexibilidade para qualquer pacote.

5.4.3 - Composição dos Diretórios

/ --- O Diretório Raiz

A composição do diretório raiz de um sistema Linux típico pode ser representado pela Tabela 01.

Tabela 01

bin	Arquivos executáveis(binários) de comandos essenciais pertencentes ao sistema e que são usados com freqüência.
boot	arquivos estáticos de boot de inicialização(boot-loader)
dev	Arquivos de dispositivos de entrada/saída
etc	Configuração do sistema da máquina local com arquivos diversos para a administração de sistema.
home	Diretórios local(home) dos usuários.
lib	Arquivos da bibliotecas compartilhadas usados com freqüência
mnt	Ponto de montagem de partição temporários
root	Diretório local do superusuário (root)
sbin	Arquivos de sistema essenciais
tmp	Arquivos temporários gerados por alguns utilitários
usr	Todos os arquivos de usuários devem estar aqui (segunda maior hierarquia)
var	Informação variável

Cada diretório listado será discutido em detalhe em uma sub-sessão separada mas adiante. /usr e /var, cada un tem seu própria sessão de documentos.

O kern do LINUX estaria localizado na raiz / ou no /boot. Se estiver localizado em / é recomendado usar O nome VMLINUX o VMLINUZ, nome que deverá ser usados em programas fonte do kern do LINUX recentemente. Mais informação da localização do kern pode-se encontrar na sessão sobre a raiz / neste trabalho.

5.4.3.1 - Subdiretório /bin

- Composição: Arquivos Binários de comandos essenciais de usuários (disponíveis para todos os usuários).

Contém os comandos que podem ser utilizados pelos usuários e pelo administrador do sistema, porém que são requeridos no modo mono-usuário (single-user mode) pode também conter comandos que são utilizados indiretamente por alguns scripts.

Todos os arquivos utilizados somente pelo root, tal como **daemons, init, gette, update**, etc. Estariam localizados em /sbin ou /usr/sbin dependendo se são ou não essenciais. Não abra subdiretórios dentro /bin.

Os arquivos dos comandos que não são suficientemente essenciais para estar em /bin estarão localizados em /usr/bin, os elementos que são utilizados pelos usuários isoladamente (porém não pelo root) (**mail, chsh, etc**.) não são suficientemente essenciais para estar dentro da partição /.

5.4.3.1.1 - Arquivos e/ou comandos disponíveis em bin

Os arquivos que devem estar em **/bin** são comandos gerais e comandos de rede.

☑ Comandos gerais:

Os seguintes comandos deverão sido incluídos porque são essenciais. Alguns estarão presente e tradicionalmente deverão estado em /bin.

- {**arch, cat, chgrp, chmod, chown, cp, date, dd, df, dmeg, echo, ed, false, kill, in, login, mkdir, mknod, more, mount, mv, ps, pwd, rm, rmdir, sed, setserial, sh, sfte , seu, sinc, true, umount, uname**}.

Se /bin/sh é bash, então /bin/sh seria em links simbólico a /bin/bash dado que bash se comporta diferente quando é carregado como sh ou bash. A pdksh que pode ser a /bin/sh nos discos de instalação e seria igualmente carregada a que /bin/sh faz um links simbólico a /bin/ksh. Outros links simbólicos de sistemas utilizando outros programas, então a partição / conterá os componente mínimos necessários.

Por exemplos, muitos sistemas incluiria cpio como de segunda utilidade mais é usado para reparos depois do tar. Porém jamais se espera restaurar o sistema da partição /, então estes arquivos podem ser omitidos (montar / em chip ROM, montar /usr no NFS). Se a restauração do sistema se planeja através da rede, Então FTP o TFTP (junto com todo o necessário para obter uma conexão FTP) estariam disponíveis na partição /.

Os comandos de restauração podem aparecer em /bin ou /usr/bin em sistemas LINUX diferentes.

Comandos de redes

Estes são unicamente os arquivos de rede que os usuários e o root queiram ou necessitem executar que não estejam no /usr/bin ou /usr/local/bin {**domain name**, **hostname**, **netstat**, **ping**}.

Subdiretório /boot

- Composição: arquivos estáticos do boot de inicialização (boot loader).

Este diretório contém tudo que é necessário para carregar o sistema, exceto os arquivos de configuração e o gerenciador de boot. O /boot é utilizado para qualquer coisa que se utiliza antes do kernel execute /sbin/init. Este inclui setores master de inicialização (master boot sectors) guardados, arquivos de mapa de setor e qualquer outra coisa que não é editada manualmente. Os programas necessários para consertar o boot de inicialização e capaz de carregar um arquivo (tal como o gerenciador de boot [lilo]) estaram localizados em /sbin. Os arquivos de configuração para carregar de inicialização poderiam estar localizados em /etc.

Como o exposto acima, o kern do LINUX pode estar localizado em / ou /boot, se estiver em /boot, é recomendado usar um nome mais descritivo.

5.4.3.3 - Subdiretório /dev

- Composição: arquivos de dispositivos de entrada/saída.

Este é diretório dos dispositivos. Contém um arquivo para cada dispositivo que o kern do LINUX pode suportar.

/dev também contém. um script carregado **MAKEDEV** o qual pode criar dispositivos quando necessitar. Pode conter um **MAKEDEV** local para dispositivos locais.

MAKEDEV deve fazer previsão para criar qualquer arquivo de dispositivo especial listado na lista de dispositivos suportados pelo Linux.

Os links simbólicos não devem ser distribuídos no sistemas LINUX, somente como prevê na lista de dispositivos de LINUX. Isto é porque as instalações locais seguro diferem daquelas da máquina do administrador. Além disso, um script de instalação configura links simbólicos na instalação, estes ligação seguramente não se atualizaram se houverem trocas locais no hardware. Quando utilizados responsavelmente são de bom uso.

A lista de dispositivos compatível com o LINUX, é mantida por Peter Anvin (peter.anvin@Linux.org). Estes arquivos especial de dispositivo estão disponível no endereço eletrônico da internet ftp.eggdrassiml.com, no diretório /pub/device-list.

5.4.3.4 - Subdiretório /etc

- Composição: Configuração do sistema da máquina local com arquivos diversos para a administração de sistema.

Contém arquivos e diretórios que são locais ao sistema atual. Nenhum arquivo deve ir diretamente dentro /etc. Os arquivos que nas primeiras versões encontravam-se em /etc, irá em /sbin ou /usr/sbin. Isto inclui arquivos tal como init, gette e update, arquivos como hostname que são utilizados por usuários comuns e pelo root não iriam em /sbin seriam em /bin.

Subdiretórios de /etc

Tipicamente /etc possui 2 sudiretórios :

- X11 arquivos de configuração para Ox11
- sk Esqueletos da configuração usuários

O /etc/sk é a localidade para os chamados arquivos esqueletos de usuários, que são os dados por default quando um novo usuário recebe uma conta, esta diretório pode conter subdiretórios para diferente grupos de usuários (/etc/skell/apoio, /etc/skell/usuários).

O /etc/X11 é o lugar recomendado para todos os arquivos de configuração X11 locais da máquina. Este diretórios é necessário para permitir o controle local se /usr for colocado somente para leitura. Os arquivos que devem ir neste diretório incluem **Xconfig** (e /o XF86Config) e **Xmodmap**. O /etc/X11/xdm contém os arquivos de configuração xdm. Estes são a maioria dos arquivos normalmente gravados em /usr/lib/X11/xdm; veja /var/lib/xdm, para maior informação.

5.4.3.4.1 - Arquivos e/ou comandos disponíveis em /etc

Este descrição do conteúdo é genérica, portanto não está totalmente completa, pode haver algumas variações dependendo do distribuidor do Linux ou do administrador de sistema. Os arquivos /etc são composto de:

- Arquivos gerais;
- Arquivos de rede.

Os arquivos gerais são necessários na maioria dos sistemas LINUX, tais como:

{adjtime, csh.login, disktab, fdprm, fstab, gettedefs, group, inittab, issue, ld.so.conf, lilo.conf, magic, motd, mtab, mtools, passwd, profile, psdaTabelase, securette , shells, se sog.conf, tercamp, tte te pe}

Os arquivos de Rede mais utilizados na maioria dos sistemas LINUX são: **{exports, ftpusers, gateway, hosts, host.conf, host.equiv, host.lpd, inetd.conf, networks, printcap, protocols, reolv.conf.rpc, service}**

Existe 2 modos para a instalação dos scripts de comandos "rc" os quais são chamados por init no momento de carregar, ou o modo /etc/rc.d/* etc do System V. Qualquer que seja o escolhido pode ser utilizado uma mescla dos dois.

Os sistemas com a senha de passwords sombreadas (shadow password) terão arquivos de configuração adicionais, em /etc (/etc/shadow e outros) e /usr/bin (useradd, usermod, e outros).

5.4.3.5 - Subdiretório /home

- Composição: diretórios locais dos usuários (opcional)

O sudiretório /home é claramente um sistema de arquivos específico do diretório local. A regra de criá-lo difere de máquina para máquina. Descreve uma localização sugerida para os diretórios local dos usuários, assim, recomendamos que todas as distribuições LINUX usem esta lugar como localização default dos diretórios locais.

Em sistemas pequenos, cada diretório de usuário é um dos subdiretórios debaixo /home, /home/dirson, /home/raulison, /home/weslei, etc.

Em sistemas grande (especialmente quando os diretórios /home são compartilhados entre várias máquinas usando NFS) é útil subdividir os diretórios local. A subdivisão pode ser implementada utilizando subdiretórios tal como /home/apoio, /home/docs, /home/cartas, etc. Muitas pessoas preferem colocar as contas dos usuários numa variedade de lugares, portanto, ninguém deverá confiar nesta localização. Se você deseja encontrar o diretório local de qualquer usuário, deveria usar a função de biblioteca **getpwent** em vez de contar com /etc/passwd, porque a informação pode estar armazenada remotamente usando sistemas como NIS.

5.4.3.6 - Sudiretório /lib

- Composição: Bibliotecas compartilhadas e módulos do kern essenciais.

O diretório /lib contém aquelas bibliotecas compartilhadas que são necessária para carregar o sistema e executar os comandos do sistema de arquivos raiz.

Módulos → Módulos de kern carregavéis.

Estes incluem em /lib/libc.so.*, /lib/libm.so.*, O linkador dinâmico compartilhado /lib/ld.so.*, e outras bibliloteca compartilhadas requeridas por arquivos em /bin e /sbin.

As bibliotecas que são necessárias somente pelos arquivos /usr (como qualquer arquivo X Windows) não pertencem a /lib. Só as bibliloteca compartilhadas requeridas para executar os arquivos dentro de /bin e /sbin devem estar ali. A biblioteca libm.so.* poderia estar localizada em /usr/lib se não é utilizada de nenhuma forma em /bin ou /sbin.

Por razão de compatibilidade, /lib/cpp necessita existir como uma referência ao pre-processador C instalado no sistema. A localização usual do arquivo é /usr/lib/gcc-lib/ <target>/<versão>/cpp. Pode existir links /lib/cpp apontando para estes arquivo ou a qualquer outra referência a ele que exista no sistema de arquivos. (Por exemplo, /usr/bin/cpp é utilizado frequentemente). A especificação para /lib/module ainda não foi definida, pois ainda não há um consenso na comunidade Linux.

5.4.3.7 - Subdiretório /mnt

- Composição: Utilizados para armazenamento de arquivos montados temporariamente.

Este diretório foi previsto para o administrador poder montar temporariamente sistemas de arquivos quando necessitar. O conteúdo deste diretório é um assunto local e não deve afetar a maneira que executamos nenhum

programa. É recomendamos a não utilização deste diretório para programas de instalação, e sugerimos utilizar um diretório temporário adequado que não esta em uso pelo sistema.

5.4.3.8 - Subdiretório /proc

- Composição: Sistema de arquivos virtual de informação de processos do kernel.

O sistema de arquivos proc é utilizado para manipular informação de processos e de sistema em vez de /dev/kmem e outros métodos similares. É recomendado a utilização deste para o armazenamento e obtenção de informação de processos, assim como outras informação do kern ou da memória.

5.4.3.9 - Subdiretório /root (opcional)

- Composição: Diretório local do superusuário (root)

O diretório / é tradicionalmente o diretório local do usuário root nos sistemas UNIX. /root utiliza-se em muitos sistemas LINUX e em alguns sistemas UNIX. O diretório local da conta do usuário root pode ser determinada por preferências locais. As possibilidades óbvias inclue em /, /root, e /home/root. Se o diretório local do root não esta armazenado na partição raiz, será necessário assegurar-se que tome / por default caso não seja localizado.

Não é recomendado o uso da conta root para coisas corriqueiras tal como ler o e-mail e ver as notícias (mail & news), recomendá-se que seja usada somente para a administração do sistema. Por esta razão recomendamos que não apareçam subdiretórios como Mail e News no diretório local da conta do usuário root. É recomendado que o mail para root seja redirecionados a um usuário mais adequado.

5.4.3.10 - Subdiretório /sbin:

- Composição: arquivos de Sistema (algumas vezes mantidos em /etc)

Os utilitários usados pela administração do sistema (e outros comandos que somente o root utiliza) estão armazenados em /sbin, /usr/sbin, e /usr/local/sbin. /sbin, tipicamente contém arquivos essenciais para dar boot ao sistema, além dos arquivos em /bin. Qualquer coisa que se executar depois sabendo que /usr foi montado (quando não há problemas) deveria estar em /usr/sbin. Os arquivos da administração do sistema root local devem estar em /usr/local/sbin.

Decidir que arquivos vão no diretório de /sbin é difícil. Se o usuário necessitar executá-lo, deve de ir para outro diretório. Se somente o administrador do sistema ou o root necessitem executar, tais como scripts da administração, então deve ir em /sbin (não /usr/sbin ou /usr/local/sbin, se o arquivo não é vital para a operação do sistema).

Arquivos como **chfn** que os usuários usam ocasionalmente devem estar em /usr/bin. **ping** que é absolutamente necessário para o root é também frequentemente usado pelos usuários, deverão estar em /bin. Os usuários comuns não terão que por os diretórios sbin em seu caminho (path).

É recomendado que os usuários tenham permissão de leitura e execução em tudo que se encontra em /sbin exceto talvez certos programas; **setuid** e **setgid**. A divisão entre /sbin e /bin não foi criada por motivos de segurança para evitar que os usuários violem o sistema operacional, foi cirada para promover uma boa partição entre arquivos que todos usam e os que são utilizados principalmente para as tarefas administrativas. Não há utilidade inerente na segurança em fazer que /sbin esteja fora do alcance dos usuários.

5.4.3.10.1 - Arquivos e/ou comandos armazenados em /sbin

Arquivos armazenados são dos seguintes tipos:

- comandos gerais;
- comandos de saída;
- comandos de manipular sistema de arquivos;

- gerenciador de boot de inicialização e
- comandos de rede.

Os comandos gerais são **clock, gette, init, update, mkswap, swapon, swapoff, telinit.**

Os comandos de saída são **fastboot, fasthalt, halt, reboot, shutdown.**

Os comandos de manipular sistemas de arquivos são **fdisk, fsck, fsck.*, mkfs, mkfs.*,** onde * = é um dos seguinte. **ext, ext2 minix, msdos, xia,** e talvez outros.

Os comandos do sistema de arquivos ext2 (opcional) são **badbocks, dumpe2fs, e2fsck, mke2fs, mkost+found, tune2fs.** O Gerenciador do boot de inicialização (lilo) e os comandos de Rede : **arp, ifconfig, route.**

5.4.3.10.2 - Arquivos opcionais em /sbin

Os arquivos estáticos (compilados estaticamente) estão o **sln** e **nc** estático, **nc** são utilitários quando ocorrem erros. O principal uso do **sln** (reparar links simbólicos incorretos em /lib depois de uma atualização mal sucedidas) não é a preocupação maior, já que existe o programa **ldconfig** (usualmente localizado em /usr/sbin) e pode atuar como um assistente guiando para atualizar as bibliotecas dinâmicas. O **nc** estático é útil em algumas ocasiões de emergência. Note que estes não necessitam ser obrigatoriamente versões compiladas estaticamente dos **ln** e **nc**, porém podem ser compilados estaticamente.

O arquivo **ldconfig** é opcional em /sbin, dado que um usuário pode escolher executar **ldconfig** ao dar boot, em vez de só quando atualizam as bibliotecas compartilhadas. (Não está claro se é ou não vantajoso executar **ldconfig** em cada inicialização). Assim, alguns que gostam de ter **ldconfig** a mão na situação que se tem um **sln**, pois não se sabe como nomear os links.

5.4.3.11- Subdiretório /tmp

- Composição: arquivos temporários gerados por alguns arquivos utilitários.

O /tmp é utilizado para arquivos temporários, preferencialmente em dispositivo rápido (um sistema de arquivos basado em memória por exemplo). A "permanência" da informação que é armazenada em /tmp é diferente de aquela que é armazenada em /var/tmp. /tmp pode ser limpo em cada inicialização ou a intervalos relativamente freqüentemente. Portanto, não se deve operar a informação armazenada em /tmp permanecendo por algum período determinado de tempo.

Os programas devem utilizar /tmp ou /var/tmp (que era originalmente /usr/tmp) de acordo os requisitos esperados da informação, pois não devem colocar nenhum arquivo particular em qualquer diretório de armazenamento temporário.

Os administradores de sistemas podem querer ajuntar /tmp a algum outro diretório, tal como /var/tmp. Isto é útil, por exemplo, para conservar espaço na partição raiz. Se está é executada, então a permanência de arquivos em /var/tmp deve ser mesmo tão grande como a de /tmp. O subdiretório /tmp pode estar na memória RAM, /var/tmp nunca poderá se localizar-se em algum dispositivo RAM.

5.4.3.12 - A hierárquia /usr.

O subdiretório /usr é a segunda maior seção do sistema de arquivos. /usr é informação compartilhada, somente de leitura, isto significa que /usr, deve ser compartilhada entre várias máquinas que utilizam o LINUX e não deve exibir qualquer informação local de uma máquina ou que varia com o tempo, deve ser armazenada em outro lugar.

Nenhum pacote grande (como **TeX** o **GNUEmacs**) deve utilizar o subdiretório direto abaixo de /usr, em vez disso, deve haver um subdiretório dentro /usr/lib (o /usr/local/lib caso tenha sido instalado localmente), a propósito, como o sistema **X Windows** faz-se uma exceção permitindo um considerável precedente e a prática amplamente aceita. Exemplo de um subdiretório /usr típico:

```
carvalho:/usr$ ls
X11@            etc/            lib/            spool@
X11R6/          games/          local/          src/
X386@           i486-Linux/     man/            tclX/
adm@            i486-Linuxaout/ openwin/        tkX/
bin/            i486-sysv4/     preserve@       tmp@
dict/           include/        sbin/
doc/            info/           share/
carvalho:/usr$
```

5.4.3.12.1 - Subdiretórios /usr (permanente)

Em um sistema típico teremos mais ou menos os seguintes diretório:

X11R6	Sistema X Windows Versão 11 release 6
X386	Sistema X Windowss Versão 11 release 5 em plataformas X 386
bin	A maioria dos comandos de usuário
dict	Listas de palavras
doc	Documentação miscelânia
etc	Configuração do Sistema
games	Jogos e arquivos educacionais
include	arquivos header(cabeçalhos) incluidos por programas C
info	Diretório primário, o sistema GNU Info
lib	Bibliotecas
local	Hierárquia local
man	Manual on line
sbir	Arquivos de Administração do Sistema não vitais
share	Informação independente da arquitetura
src	Código fonte

Os seguinte links simbólicos a diretórios podem estar presentes. Esta possibilidade baseia-se na necessidade de preservar a compatibilidade com sistemas anteriores haja visto que em todas as implementações pode assumir o uso da hierarquia /var, poderão existir os seguintes links :

/usr/adm	→	/var/adm
/usr/préerve	→	/var/préerve
/usr/spool	→	/var/spool
/usr/tmp	→	/var/tmp
/var/spool/locks	→	/var/lock

Uma vez que o sistema não precise mais de alguns dos liks anteriores, deve existir um links deste /usr/X11 apontando para a hierarquia do sistema X Windows atual.

5.4.3.12.2 - Subdiretório /usr/X386

É composta do sistema X Windows, versão 11 release 5 em plataformas X 86, esta hierarquia é geralmente idêntica a /usr/X11R6, exceto que os links simbólicos de /usr devem estar ausente se estiver instalado /usr/X11R6.

5.4.3.12.3 - Subdiretório /usr/bin

É composta da maioria dos comandos do usuário, este é o diretório principal de comandos executáveis no sistema possui o **mh** (comandos para o sistema de manipular e-mail MH) e o **X11** (link simbólico até /usr/X11R6/bin).

Os interpretadores de scripts dos shell (invocados com **#! <rota>** na primeira linha do script de shell) não podem depender de uma rota, é vantajoso o padronizar a localização dos elos. A shell Bourne e C é tão fixos em **/bin**, pois **Perl, Pethon, Tlc** se encontram em muitos lugares diferentes **/usr/bin/perl, /usr/bin/pethon e /usr/bin/tcl** devem referênciar a os interpretador de shell

perl, pethon e tcl respectivamente. Estes podem ser links simbólicos a localização física dos interpretador da shell.

5.4.3.12.4 - Subdiretório /usr/dict - Listas de palavras

Arquivos recomendados em /usr/dict (words), tradicionalmente esta diretório contém somente arquivos words de palavras inglesas, o qual é utilizado por "look" para vários programas de ortografia, words pode utilizar ortografia americana ou britânica. Os usuários que precisam ambos, podem concatenar words a /usr/dict/american-english ou /usr/dict/ british-english.

As listas de palavras para outros linguagem pode usar o nome em inglês para a linguagem, por exemplo, /usr/dict/french, /usr/dict/danish, etc. Estes devem, se possível, utilizar o jogos de caracteres ISO 8859 que faz aprópriado para linguagem em questão, e se possível, o jogo de caracteres ISO 8859-1 (Atin1) deve ser utilizado (muito raramente é possível fazê-lo)

Qualquer outra lista de palavras, tal como o diretório **web2**, deve ser incluido aqui, é razoável ter aqui só as listas de palavras, e que elas são os únicos arquivos comum a todos os verificadores de ortografia.

5.4.3.12.5 - Subdiretório /usr/etc

Contém a configuração do sistema, porém armazenar a configuração /usr/etc do software que se encontra em /usr/bin e /usr/sbin é um problema. Montar /usr somente para leitura de um CD-ROM ou através de NFS é difícil no melhor dos casos.

Uma possível solução que considerada foi eliminar completamente /usr/etc e especificar que todas as configurações se armazenem em /etc. Acontece que existe a possibilidade de que muitos site podem querer ter alguns arquivos de configuração que não estejam na sua máquina local.

Eventualmente, decide-se que /etc deverá ser o único diretório que seja referenciado pelos programas (Isto é, todos devem buscar configurações em /etc e não /usr/etc). Qualquer arquivo de configuração que necessissário para todo o sistema e que não era necessário antes de montar /usr (em uma situação de emergência deve estar localizado em /usr/etc). Arquivos especificos em /etc, em máquinas específicas podem ter ou não um link simbólicos aos arquivos de configuração localizados em /usr/etc. Isto também significa que /usr/etc é tecnicamente um diretório opcional no sentido restrito, mesmo assim recomendamos que todos os sistemas LINUX o incorporem.

Não é recomendado que /usr/etc contenha links simbólicos que apontem para arquivos /etc. Isto é desnecessário e interferem no controle local das máquinas que compartem o diretório/usr.

5.4.3.12.6 - Subdiretório /usr/include

Neste diretório é onde todos os arquivos include de uso geral do sistema para programação em linguagem C e C++ podem ser localizados. Descrição dos principais sudiretórios de /usr/include: /usr/include arquivos include

X11	Link simbólico até /usr/X11R6/include/X11
arpa	Definição do protocolo definido por ARPNET.
asm	Link simbólico até /usr/scr/Linux/include/asm-<arch>.
bsd	arquivos include de compatibilidade com BSD.
g++	arquivos include de GNU C++.
gnu	arquivos include GNU.
Linux	Link simbólico a /usr/src/Linux/include/Linux.
net	Definição genéricas relacionadas com rede.
netax25	Definição específicas a +AX25 (ARRL AX25).
Netinet	Definição específicas a TCP/IP.
netipx	Definição específicas a +IPX (NovOIPX/SPX).
protocols	Definição de protocolos(basadas em INET)
readline	A bibliloteca readline GNU.
rpc	Definição RPC de Sun Microsystem.
Rpcsvc	Definição de serviços RPC de Sun Microsystem.
sys	arquivos include de geração de sistemas

O subdiretório **arpa** contém definições de header de protocolos para os protocolos **ARPANET, TCP/IP**, definições para ftp, prototipos telnet e material similar.

O subdiretório **net** contém definições genéricas relacionadas com a rede, define a interface sistema vs. kern, detalhes da família de protocolo, etc.

O subdiretório **netinet** contém definições específicas de INET (DARPA Internet, que também é contida no TCP/IP).

ARRL AX.25 é melhor conhecido como pacote de transmissão via radio (packet radio). Os protocolos novell **IPX/SPX** são parte dos serviços de arquivos Novell Netware.

5.4.3.12.7 - Subdiretório /usr/lib

Inclui as bibliloteca para programas e pacotes, inclue as bibliloteca objeto, arquivos de programa compilador, informação estática de várias casos, ambos, códigos executável (por exemplo os arquivos internos de **gcc** estão localizados abaixo **/usr/lib/gcc-lib**) e outros tipos de informação. /usr/lib/ - bibliloteca para programação e pacotes:

X11	Link simbólico para /usr/X11R6/lib/X11
emacs	arquivos de suporte estáticos para o editor GNUEmacs.
games	arquivos de dados estáticos para /usr/games.
groff	Bibliloteca / diretórios para GNU groff
gcc-lib	arquivos/diretórios especificos do sistema para gcc.
kbd	Tabelas de tradução de teclado e informação relacionada.
Mh	Bibliloteca para o sistema de manipular e-mail MH:
news	Cnews/INN.
smail	Smail.
terminfo	diretórios para a base de dados terminfo.
texmf	TeX/MF (e ATeX) bibliloteca de informação.
uucp	Comandos de UUCP.
zoneinfo	Configuração e informação da zona horaria.

Historicamente, /usr/lib é incluido além disso alguns comandos executáveis tais como **sendmail** e **makewhatis**.

Dado que **makewhatis** não é referênciado por outros programas, não há problemas ao mover para um diretório de arquivos executáveis. Arquivos que os usuários precisam para usar **makewhatis**, /usr/lib de onde pertencem. O arquivo **catman** que repassa ao script **makewhatis** em muitos sistemas LINUX, deve também estar em usr/bin.

O arquivo **sendmail** é referênciado por muitos programas com seu nome histórico /usr/lib/sendmail. Este deve ser um links simbólico, a localização layout para os agente de transferência de e-mail com uma interface de linha de comando compatível com o sendmail, /usr/bin/sendmail.

Em sistemas que utilizam smail devem localizar smail em /usr/sbin/smail e /usr/bin/sendmail deve ser um links simbólico a smail.

Este regra vai de encontro também com a nova ocorrência no layout sendmail definida em Sendmail 8.6.x e BSD 4.4. Note que esta localização requer que /usr/sbin e /usr/sbin/sendmail devem ser executáveis para usuários normais. Qualquer pacote de programa que contenha e precisa informação que não necessite ser modificada deve armazenar tal informação em /usr/lib (o /usr/local/lib, esta instalado localmente). Recomenda-se a utilização de um subdiretório em /usr/lib para este propósito.

A informação de jogos armazenada em /usr/lib/games deve ser apenas informação estática. Qualquer arquivo modificável tal como arquivos demarcado, registros de jogos e similar, devem de ser localizados em var/lib. É necessário para compatibilidade de jogos, pode-se usar um links simbólico desde /usr/games/lib até /usr/lib/games.

Nota: Nenhuma informação especifica de host para o sistema X Windows deve armazenar-se em /usr/lib/X11 (que é realmente /usr/X11R6/lib/X11). Os arquivos de configuração especificos do host tal como **Xconfig** o **XF86Config** devem ser armazenados em /etc/X11. Este deve incluir informação de configuração como o stem.twmrc, se for somente um links simbólico, um arquivo de configuração mais global (talvez em /usr/etc/X11 ou /usr/X11R6/lib/X11).

5.4.3.12.8 - Subdiretório /usr/local

A hierarquia /usr/local está para ser utilizada pelo administrador de sistemas quando instala o Linux localmente. Necesssita ficar a salvo de ser sobrescrito quando o software do sistema se atualiza. Pode ser usado por programas e por informação que são compatível entre um grupo máquinas, pois não se encontram em /usr. /usr/local Diretórios da Hierárquia local:

bin	arquivos
doc	Documentação local
etc	arquivos de configuração utilizados somente no local
games	Jogos instalados localmente
lib	Bibliloteca s para /usr/local
info	Páginas de informação local
man	Hierárquias de páginas de manual para /usr/local
sbin	Administração do sistema
scr	Código fonte local.

Este diretório deve estar vazio ao terminar de instalar LINUX pela primeira vez. Não deve haver exceções regra, exceto quiça os subdiretórios vazios listados. O software instalado localmente deve estar localizado dentro de /usr/local, em vez de /usr a menos que esteja sendo instalado para reemplantar ou atualizar software em /usr.

Note que o software localizado em / ou em /usr pode ser sobrescrito por atualizações do sistema (assim mesmo, é recomendado que as distribuições não sobrescrevam informações /etc fora destas circunstâncias). Por esta razão, o software local não deve se colocado fora de /usr/local sem uma boa causa.

5.4.3.12.9 - Subdiretório /usr/man

Inclui as paginas do manual, detalha a organização das páginas do manual através do sistema, devem estar dentro de /usr/man. As páginas do manual estão armazenadas <mandir>/<locais>/man [1-9]. Faremos uma pequena listagem de <mandir> e <locais>: <mandir>/<locais> uma hierarquia de páginas de manual.

man1	Programas para usuários.
man2	Chamadas do sistema.
man3	Subrotinas e funções de bibliloteca.
man4	Dispositivos.
man5	Formatos arquivos.
man6	Jogos.
man7	Misceláneas.
man8	Administração do Sistema.
man9	Funções e variáveis internas do kernel.

O <mandir> primário do sistema é /usr/man contém informação do manual para comandos e informação abaixo dos sistemas de arquivos / e /usr. Obviamente não há páginas de manual em / porque não se necessitam para carregar nas emergências. Deve-se fazer reserva na estrutura de /usr/man para o suporte de páginas do manual que estão escritas em diferentes idiomas (multiplos idiomas). Estas providências devem levar em conta o armazenamento e referência destas páginas do manual. Os fatores relevantes incluir no idioma (inclue diferenças basedas na geografia) e código do conjunto caracteres. Esta nomenclatura dos subdiretórios de idiomas de /usr/man esta basada no apêndice e do padrão POSIX

1003.1 que descreve a cadeia de identificação locais. O método mas aceito para descrever um ambiente cultural. A cadeia <locais> é:

<idioma>[<_territorio>][.<conjunto_de_caracteres>][,<versão>]

O campo <idioma> vem do ISO639 (um código para a representação dos nomes dos idiomas). Seja os caracteres especificado no padrão ISO, com minúsculas somente.

O campo <_territorio> será o código das letras de ISO3116 (uma especificação da representação dos nomes dos países, se possível (muita gente está familiarizada com o código 2 letras espelhado no código país como e-mail).

O campo <conjunto_de_caracteres> deve representar o layout que descreve o código caracteres. Se o campo <conjunto_de_caracteres> é só uma especificação numérica, o número representa o número do layout internacional que descreve o conjunto caracteres. Recomenda-se que utilizar uma representação numérica, sempre que for possível (especialmente o padrão ISO), que não inclua símbolos de pontuação e que todas as letras sejam minúsculas.

Um parâmetro que especifique <versão> do perfil pode ser colocada depois do campo <conjunto_de_caracteres>. Esta pode utilizar-se para diferenciar as necessidade culturais.

Em sistemas que usem só um idioma e um código do conjunto de caracteres para todas as páginas do manual, pode-se omitir a subcadeia <locais> e armazenar todas as páginas do manual em <mandir>. Por exemplo no sistemas que só tem páginas do manual em inglês codificados na ASCII, podem armazenar as páginas do manual (Os diretórios man[1-9]) diretamente em /usr/man.

Em países nos quais existe um código do conjunto caracteres no layout, pode omitir o campo <conjunto_de_caracteres>, porém é bastante recomendado que a inclua, especialmente para países com vários layouts. Exemplos de vários manuais encontrados:

Idioma	Países	Conjunto caracteres	Diretório
Inglês	--------	ASCII	/usr/man/em
Inglês	Reino Unido	ASCII	/usr/man/em_GB
Inglês	Estados Unidos	ASCII	/usr/man/em_US
Francês	Canadá	ISO8859-1	/usr/man/fr_CA
Francês	França	ISO8859-1	/usr/man/fr_FR
Alemão	Alemanha	ISO646-DE	/usr/man/de_DE646de
Alemão	Alemanha	ISO6937	/usr/man/de_DE6937
Alemão	Alemanha	ISO8859-1	/usr/man/de_DE.88591
Alemão	Suiça	ISO646-CH	/usr/man/de_CH.646ch
Japonês	Japão	JIS	/usr/man/ja_JP.jis
Japonês	Japão	SJCS	/usr/man/ja_JP.sjis
Japonês	Japão	UJ (ó EUC-J)	/usr/man/ja_JP.ujis

As páginas do manual para os comandos e informação que se encontra abaixo /usr/local estão armazenadas em **/usr/local/man**. As páginas do manual para o sistema **X Windows** estão armazenadas em **/usr/X11R6/man**. Logo todas as hierarquias de páginas do manual no sistema devem ter a mesma estrutura que /usr/man. Os diretórios vazios podem ser omitidos da hierarquia de páginas do manual. Por exemplo se, /usr/local/man não tem páginas do manual na seção 4 (dispositivos) então se pode omitir /usr/local/man/man4.

As seções da páginas cat(cat[1-9]) que contém **páginas do manual formatadas**, também se encontram dentro os subdiretórios /<mandir>/<locais>, pois não são requeridas nem devem ser distribuidas no lugar das fonte **nroff** das páginas do manual.

As páginas do Manual do sistema de manipulação de e-mail **mh** devem ter o prefixo **mh** em todos os nomes de arquivos das páginas.

As páginas do sistema **X Windows** devem de ter o prefixo **X** em todos os nomes dos arquivos das páginas. A prática de colocar as páginas do manual de diferentes idiomas, nos subdiretórios apropriados de /usr/man também se aplica a as outras hierarquias de páginas do manual, tais como /usr/local/man e /usr/X11R6/man. Isto também é aplicável a estrututura opcional de /var/catman, mostrada no subdiretório /var.

5.4.3.12.10 - Subdiretório /usr/sbin

Este diretório contém quaisquer arquivo não essenciais utilizado exclusivamente pelo administrador do sistema. Os programas de administração do sistema que sejam utilizados para a reparação do sistema, montado no /usr, outras funções essenciais devem localizar-se em /sbin em vez de /usr/bin.

Tipicamente /usr/sbin contém os **deamons** de rede, quaisquer ferramenta de administração não essenciais e arquivos para programas servidores não-críticos. Estes inclue os **deamons** da internet que são chamados por inted (chamados in.*) tais como **in.telnetd** e **in.fingerd** e os deamons basados em **rpc** manipulados por **portmap** (chamados rcp.*), tais como **rcp.infsd e rcp.mountd**.

Estes programas servidores são utilizados quandoocorre um estado que o System V conhece como "run levO2" (estado multi-usuário) e o "run levO3" (estado em rede) ou estado que o BSD conhece como "modo multi-usuário", neste ponto ficam disponíveis os serviços para os usuários (suporte de impressão) e até outras máquinas (por exemplo, exportar NFS). Os programas administrativos instalados localmente devem estar localizados em: /usr/local/sbin.

5.4.3.12.11 - Subdiretório /usr/share

São informação que independente da arquitetura, quaisquer especificação para /usr/share será incluida em um documento suplementar ao FSSTND, de acordo com a Linux Organization, os pesquisadores do FSSTND acham que /usr/share não é necessário na maioria dos sistemas Linux.

5.4.3.12.12 - Subdiretório /usr/src

Contém o Código fonte para o kern do Linux, qualquer código fonte não local deve localizar-se neste diretório. O único código fonte que sempre deve localizar-se em um lugar específicos é o código do kern(quando exista ou esteja enlaçado como parte de uma estrutura /usr/include). Podem-se usar subdiretórios que desejar. O código fonte para Kern deve sempre estar em seu lugar mesmos. Os arquivos include do código do kernel. Esses arquivos estão localizados neste diretórios.

/usr/src/Linux/include/asm-<arch>
/usr/src/Linux/include/Linux

/usr/include deve conter links a estes diretórios, chamados asm e Linux, dados que são necessitados pelo compilador de C, ao menos estes arquivos include devem sempre ser distribuidos nas intalações que incluem um compilador C. Devem ser distribuidos no diretório/usr/src/Linux de forma que não existão problemas quanto os administradores do sistema atualizem sua versão do kern pela primeira vez.

/usr/src/Linux pode também ser um links simbólico a um árvore de código fonte do kernel.

5.4.3.13 - A Hierárquia /var

/var	Informação variável
adm	Informações administrativa do sistema (obsoleto). Link simbólico até
	/var/og
catman	Páginas do manual formatadas localmente
lib	Informação do estado das aplicações
local	Informação variável do software de /usr/local
ock	arquivos de bloqueio
og	arquivos de Agenda
named	arquivos DNS, somente rede
nis	arquivos base de dados NIS
run	arquivos relevantes a processos execução do sistema
spool	Diretórios de trabalhos em fila para realizar-se depois
tmp	arquivos temporários, utilizado para manter /tmp menor possível

/var contém arquivos com informação variável. Esta incluem arquivos e diretórios em fila de execução, informação de ordem administrativa e arquivos temporários e transitorios.

Algumas porção de /var são não compatível entre diferentes sistemas. Por exemplo, /var/og, /var/ock e /var/run. Outras porção são compatível, notoriamente /var/spool/mail e /var/spool/news.

/var é especificada aqui para fazer possível montar /usr somente para leitura. Tudo aquilo que alguma vez ficou em /usr é escrito durante a operação normal do sistema (não durante a instalação e manutenção do software) deve ir em /var.

Se /var não pode ser uma participação separada, é preferivel mover /var para fora do diretório raiz, pois dentro da partição /usr (Isto se fazia algumas vezes para reduzir o tamanho da partição raiz ou quando há pouco espaço na partição raiz). Sendo assim, /var não deve ser enlaçada a /usr, porque fazia que a separação entre /usr e /var seja mais difícil e seguramente criará um conflito do nomes, em vez links /var e / usr/var.

5.4.3.13.1 - /var/adm : Agenda do sistema e arquivos contabilizados *(obsoleto)*

Este diretório tem sido repassado para /var/og e outros diretórios. Deve ser um links simbólico a /var/og até que todos os programas não se refiram mas a algum arquivo em /var/adm. utmp seja movido a /var/run. Todos os arquivos agendas vão ser movidos a /var/og incluindo o arquivo wtmp. O suporte de empacotamento das distribuições deve armazenar em /var/lib/<nome>.

Nota: O links simbólico /var/adm não deve ser necessário a maioria dos sistemas Linux-i386ELF dado que Otroca foi introduzido antes que ELF fora liberado ao público.

5.4.3.13.2 - /var/catman : Páginas do Manual Formatadas localmente (opcional)

Este diretório poporcionara uma localização padrão para os computadores que utilizam uma partição /usr somente para leitura, pois desejam permitir o armazenamento temporário de páginas do manual formateados localmente. Os administradores que montaram /usr como escrita (intalações mono-usuários) podem escolher não usar /var/catman e exibir as páginas do manual formatadas dentro dos diretórios cat[1-9] dentro /usr diretamente. Recomendamos que a maioria dos administradores utilizem uma das seguintes opções em seu lugar.

Preformatando todas as páginas do manual dentro /usr com o programa (catman). Não será permido o armazenamento temporário das páginas formatadas do manual e precise que se execute **nroff** cada vez que necessite uma página.

Se permita o armazenamento temporário local das páginas do manual em /var/catman.

A estrutura de /var/catman necessita refrear ambos, o trecho da existência de múltiplas hierarquias de página do manual e a possibilidade do uso de multiplos idiomas.

Dada uma página do manual sem formatar que normalmente aparece em /usr/<rota1>/man/man[1-9], a versão formatada armazenada temporáriamente deve ir em /var/catman/<rota2>/cat[1-9], aonde <rota2> é <rota1>. Os componentes <rota2> e <rota1> estão ausente no caso de /usr/man e /var/catman.

Por exemplo, /usr/man/man1/ls.1 é formatado em /var/catman/cat1/ls.1 e /usr/X11R6/man/<locais>/man3/XtCass.3x esta formatado em /var/catman/X11R6/ <locais>/cat3/XtCass.3x.

As páginas do manual escritas em /var/catman/cat[1-9] podem eventualmente, transferir-se a /usr/<rota>/cat[1-9]. De igual forma as páginas do manual formatadas dentro de /usr/<rota>/cat[1-9] podem expirar se não são acceitas num período do tempo.

Se tivessem páginas do manual preformatadas com um sistema Linux num meio somente para leitura (por exenplo um CD-ROM), devem estar instaladas em /usr/<rota>/cat[1-9]. /var/catman está reenviado para um lugar de armazenamento temporário para páginas de manual formatados.

5.4.3.13.3 - /var/lib: Informação de Estado das Aplicações.

/var/lib.- Informação de Estado das Aplicações

emacs	Diretório do estado do Emacs
games	Informação variável de jogos
news	Arquivos variáveis de Cnews/INN
texmf	Informação variável asociada com TeX
xdm	arquivos de autenticação e código de erros de aplicações X Windows.

/var/lib/<nome> é o lugar apropriado para o suporte de empacotamento de todas as distribuições. Diferentes distribuições de Linux podem utilizar diferentes nomes para suporte.

/var/lib/emacs

O diretório do estado GNU Emacs, O lugar do **ndos** arquivos de informação independente da arquitetura, que Emacs modifica quando executa, deve ser /var/lib. No presente, Emacs somente localiza seu diretório de arquivos de bloqueio abaixo do diretório de estado (em <direstado >/emacs/lock), pois pode fazer uso mais extenso do mesmo no futuro, notoriamente, só requer a adição de uma opção sensível no programa configure de Emacs para fazer esta troca (antes de compilar).

/var/lib/games

Assim como os subdiretórios antes citados, quaisquer informação variável relacionada com os jogos que se encontram em /usr/games, devem estar aqui. /var/lib/games deve incluir a informação variável que previamente será encontrada em /usr/lib/games. A informação estática, tal como textos de ajuda, descrições do niíveis devem permanecer em /usr/lib/games.

/var/lib/news

/var/lib/news deve usar para armazenar toda a informação variável asociada com os servidores de news tais como Cnews e INN, inclusive o arquivo histórico, o arquivo ativo.

/var/lib/texmf

/var/lib/texmf deve usar para armazenar a informação variável associada com TeX. Particularmente, em /var/lib/texmf/fonts armazenaram todas as fonte tipográficas que são geradas automáticamente por **MakeTeXPK.**

Deve haver um links desde /usr/lib/texmf/fonts/tmp até /usr/lib/texmf/fonts. Este links permite os usuários fazer uso de uma só rota /usr/lib/texmf/fonts/tfm quando houver trocas da sua variável TEXFONTS (Esta é A rota default nas ferramentas TeX de Karl Berre distribuidas por ftp.cs.umb.edu:pub/tex [A razão

de mencionar-lo aqui é que são o padrão de fato nas intalações UNIX, estas ferramentas são amplamente usadas na comunidade LINUX]. Se se utiliza outra distribução de TeX, deve fazer um links deste diretório de fonte apropriada até /usr/lib/texmf/fonts).

O **MakeTeXPK** que se distribue e com **dvipsk** colocará os arquivos .pk em fonts/pk/<dispositivo>/<nome_da_fonte>, (por exemplo, fonts/pk/Canon_CX/cmr10.300pk). Os arquivos .pk podem ser plugados periodicamente do árvore /var/lib/texmf ou pode-se mover dentro da árvore /usr/lib/texmf. Se usarem geradores automáticos de **.mf** ou **.tfm**, estes devem por sua informações nos subdiretórios **mf** ou **tfm** de /var/lib/texmf/fonts.

/var/lib/xdm

/var/lib/xdm contém a informação variável de xdm que consiste nos arquivos xdm-errors e quaisquer arquivo pertencentes a xdm. Os arquivos de xdm tais como **chooser** devem até estar localizados na localidade histórica em /usr/X11R6/lib/X11/xdm. O arquivo xdm-pid devem estar em /var/lib/xdm apesar de existir /var/run. Os arquivos restantes devem estar em /etc/X11/xdm.

5.4.3.13.4 - /var/local : Informação variável do software que está em /usr/local

Este diretório contém toda a informação variável que esta relacionada com o software que se encontra em /usr/local. Naturalmente a implementação desta subdiretório é prerrogativa do administrador do sistema. Como a informação pode estar noutro lugar do diretório/var, não deve colocar em /var/local. Por exemplo, todos os arquivos de bloqueios estarão em /var/ock.

5.4.3.13.5 - /var/ock : arquivos de Bloqueio

Os arquivos de bloqueio devem de armazenar-se dentro uma estrutura do diretório de /var/ock. Para preservar a habilidade de montar /usr somente para leitura , não se deverá colocar os arquivos de bloqueio na partição /usr.

Os arquivos de boqueio dos dispositivo, tais como os arquivos de boqueio do dispositivos série que antes se encontravam em /usr/spool/ock ou em /usr/spool/uucp devem agora ser armazenado em /var/ock. A convenção para a nomenclatura que deve utilizar-se é LCK., seguido do nome base do dispositivo. Por exemplo, para bloquear /dev/cua0 se deverá criar o arquivoLCK..cua0.

O formato usado para os arquivos de bloqueios de dispositivo no Linux deverá ser o Formatos arquivos de bloqueio HDB UUCP. O formato HDB é armazenado em OPID (Identificador de processo) com um número decimal na ASCII de 10 bytes, com um caracter de linha nova.

Por exemplo, se o processo 1230 retém um arquivo de bloqueio, contém dados seguinte onze(11) caracteres: espaço, espaço, espaço, espaço, espaço, espaço, um, dois, três, qautro e nova linha.

Então quaisquer coisa que usar /dev/cua0, pode ler o arquivo de bloqueio e atuar de acordo (todos os arquivos de bloqueio em /var/ock devem ser lidos por todos).

54.3.13.6 - /var/og : Arquivos agenda e diretórios

Este diretório contém arquivos agenda misceláneos. A maioria dos arquivos agenda se devem exibir neste diretórios ou subdiretórios aprópriados.

astog	Registro do último acesso de cada usuário
mesage	Mensagem do sistema desde que logou ao sistema
wtmp	Registro de todos os acessos e saídas

Pode requerer um links simbólico desde /var/og/utmp até /var/run/utmp basta que nenhum programa se refira a /var/adm/utmp (/var/adm é em si mesmo um links simbólico transicional até /var/og).

5.4.3.13.7 - /var/named : arquivos DNS

Este diretório contém todos os arquivos de trabalho do servidor de nomes Internet, named. Recomendamos que /etc/named.boot seja um links simbólico até

/var/named/named.boot, dado que /etc/named.boot é o arquivo de inicialização default, se não fornece argumentos a named.

5.4.3.13.8 - /var/nis

Arquivos de base de dados do serviço de informação de rede (NIS) o sistema de informação de rede (NIS) era anteriormente conhecido como as páginas Amarelas Sun. A funcionalidade e localização de diretórios de ambos é o mesmo pois o nome (yelow Page) é uma marca registrada no Reino Unido, pertencem a Bristish Telecommunications plc. e não pode ser usada sem permisão.

5.4.3.13.9 - /var/preview:arquivos guardados depois de uma colisão ou uma termino inesperado, exemplo vi (editor de texto).

Este diretório contém os arquivos que são armazenados antes de quaisquer terminação não esperada de ex. vi.

5.4.3.13.10 - /var/run : arquivos variáveis de tempo de execução

Este diretório contém arquivos com informação do sistema que o descrevem desde que inicializou. Geralmente os arquivos neste diretório devem ser deletar (remover ou truncar) aol começar o processo de inicialização.

Os arquivos identificados do processo (PID), que estavam originalmente /etc, devem colocar em /var/run. A convenção de nomenclatura dos arquivos PID é <nome-programa>.pid, por exemplo o arquivo PID de crond se chama /var/run/crond.pid.

O formato interno dos arquivos PID permanecem sem troca. O arquivo deve conter o indicador de ponto decimal codificado como ASCII, seguido por um caracter nova linha. Por exemplo, o processo número 25, /var/run/cond.pid conterá 3 caracteres, dos cinco e nova linha.

Os programas que leiam arquivos PID devem ser flexível na acepção, por exemplo devem ignorar os espaços extras, zeros a esquerda, ausência do caracter nova linha ou linhas adicionais no arquivo PID. Os programas que criam arquivos PID devem utilizar a especificação dada no parágrafo anterior.

O arquivo utmp, que armazena informação acerca de quem está atualmente utilizando o sistema, se localiza neste subdiretório.

Os programas que mantenham sockets transitorios de dominio UNIX, devem colocaá-los neste diretório.

5.4.3.13.11 - /var/spool

Diretórios de fila de trabalhos para procedimento posterior /var/spool é tradicionalmente utilizado para a informação local de máquina que é enviada para processo depois, até o subsistemas UNIX. Por exemplo, trabalhos de impressão que são armazenados aqui para entrega posterior ao daemon da impressora, o e-mail que saí é armazenado aqui para entrega a sistemas remotos e os arquivos UUCP são armazenados aqui para transmissão dos sistemas UUCP vencidos o e-mail que entra e as noticias são armazenadas aqui para entregar-se aos usuários e os trabalhos de at e cron são armazenados aqui para execução posterior pelo daemon cron.

/var/spool

at	Trabalhos de at
cron	Trabalhos de cron
lpd	Diretório de impressora *
mail	arquivos caixa-postal dos usuários
mqueue	Fila de espera dos correio
news	Diretório de noticias *
rwhod	arquivos rwhod
smail	Diretório de smail *
uucp	Diretório de UUCP

Nota: ***** Segnifica fila de trabalhos para processamento posterior.

Os arquivos de bloqueio UUCP devem localizar-se em /var/ock. Veja a seção acerca de /var/ock.

/var/spool/lpd

/var/spool/lpd Diretório de fila de trabalhos para processamento posterior a impressão
<impressora> Diretório que tem a fila especifica desta impressora

O arquivo de bloqueio para lpd, lpd.ock deve estar localizado em /var/spool/lpd. O arquivo de bloqueios de cada impressora deve localizar-se no diretório<impressora> da impressora especifica e deve chamar ock.

5.4.3.13.12 - /var/tmp : Arquivos temporários, utilizando para manter /tmp pequeno.

Os arquivos que estão no /var/tmp estão armazenados por uma duração não especifica. (Lembre-se que os diretórios temporários do sistema não garantiram manter a informação por nenhum período particular).

A informação armazenada em /var/tmp tipicamente esta numa "forma definida localmente", pois usualmente é menos frequentemente que /tmp. Se pode encontar informação sobre diretórios temporários na seleção dedicada a /tmp (acima).

Deve existir um links simbólico desde /usr/tmp até var/tmp por razão de compatibilidade.

5.4.4 - Alguns dilemas sobre o Sistema de Arquivos

A rede apresenta um dilema intirerante, algumas pessoas quiseram separar os arquivos de rede e configuração dos outros arquivos de configuração. Ou seja, estão em desacordo. Sentimos que a rede não é um "pacote", senão uma parte integral da maioria das máquinas UNIX (e similares). Não se deve colocar a rede em um só diretório senão localizar-se sistematicamente nos diretórios aprópriados.

/bin **{hostname, netstat, ping}** Qualquer coisa que algum usuário queiram utilizar considerado vital.
/sbin **{arp, ifconfig, route}** Qualquer coisa que só root necessita e considera vital.
/usr/bin **{finger, rep, rogin, telnet, etc.}** Alguns arquivos que algum usuário queira utilizar e que não são vitais.
/usr/sbin **{in.ftpd, inetd, lpd, portmap, etc..}** Alguns arquivos que somente o administrador utiliza, que não são vitais.

Pode parecer confuso a princípio (leva tempo digerindo), tem sentido . Se por alguma razão você só pode montar a partição raiz, e necessita acessar a rede para reparar seu sistema, não quer que os arquivos estão em /usr/etc (como estão algumas vezes). Os arquivos que necessitam para montar /usr as situações normais (e de emergência) estão coocados dentro da sub-árvore raiz, e quaisquer outros podem colocar em /usr, para manter o tamanho do sistema de arquivos raiz pequeno. Os arquivos de configuração para a rede pertencem a /etc.

A Estrutura independente da arquitetura, o diretório/usr/share tipicamente contém arquivos independente da arquitetura, tais como páginas do manual, fuso horário, informação de terminais, etc. No momento presente não há diferentes arquiteturas para Linux, pois com o tempo, veremos que Linux incluirá outras arquiteturas e outros sistemas similares a UNIX.

Nota: Nenhum programa nunca deverá fazer referência a alguma coisa em /usr/share. Por exemplo, um programa de páginas do manual não deve nunca buscar diretamente /usr/share/man/man1/1s.1, sempre deve referir a /usr/man/man1/1s.1. Qualquer coisa em /usr/share, será "apontada" através do uso do enlacé símbolos de outras áreas do sistema de arquivos, tais como /usr/man, /usr/lib/<algo>, etc. Até se trabalhar as especificações de /usr/share.

Os Link simbólicos, existe muitos usos para os link simbólicos em cada sistemas de arquivos. Embora este documento como esta não respalda o uso do link simbólicos na implementação default (os encontrados depois de instalar Linux), usam frequentemente com bons propósitos em diferentes sistemas. O ponto

é que os link simbólicos devem estar ali para manter todos os arquivos e diretórios onde cada usuário espera encontar.

Estar preparado para acertar que certos diretórios, até aqueles contidos no diretório raiz, até sejam link simbólicos. Por exemplo em alguns sistemas /home não estará na raiz, senão enlaçado simbolicamente a um diretório/var ou algum outro lugar. /home poderia ter também sua própria partição física e desde logo, ser montada como tal.

Similarmente, dado que /usr poderia estar em um servidor de arquivos central montado vía NFS, /usr/local pode-se enlaçar simbolicamente a /var/local. Este troca pode-se justificar recordandoa a razão principal de ter /var: separadas de diretórios de arquivos que variam com o tempo e entre diferentes sistemas e máquinas daqueles que podem compartilhar e sejam somente para leitura.

Alguns sistemas além disso enlaçar /tmp a /var/<algo> se a partição raiz se torne muito pequena (ou é muito pequena). Existe mais exemplos de bons usos de link simbólicos, pois todo o assunto não se reduz a estas coisas: os pacotes devem ser capazes de encontar as coisas onde esperam (razoavelmente) e os link simbólicos pode-se utilizar para resolver os problemas de muitos casos. Ou seja, se podem gerar problemas com o uso demasimados link simbólicos. Este problema influi sobre a confiança nos link simbólicos para resolver problemas, confusão resultante do sobre o uso do link simbólicos e as preferências estéticas das diferentes pessoas.

Os Arquivos Compilados Estáticamente o Linux executa atualmente em uma gama de sistemas, alguns com somente um usuário e disco pequeno, outros como servidores em ambiente com rede muito grande, dada esta variedade, esta documento não impõe regra sobre quais arquivos estão compilados estaticamente ou dinamicamente, com as seguinte exceções. Ambos **ln** e se **nc**, devem existir em /bin; quaisquer versão estática pode-se colocar em /sbin ou repassa-la em /bin.

Os grande sistemas Linux podem desejar incluir outros arquivos **estáticos (sh, init, mkfs, fsch, tunefs, mount, umount, swapon, swopff, gette, login e outros).** Os instaladores e os administradores de sistemas, são livres para conectar dinâmicamente ou estaticamente estes outros arquivos segundo sua conveniência, sempre que a localização dos arquivos arquivos não troque.

Em sistemas de rede, (especialmente aqueles que não tem unidade de disco flexivel), podem querer compilar estaticamente **ifconfig**, **route, hostname** e outras ferramentas de rede. Isto usualmente não é necessário.

Pontos Positivos e negativos

O Linux é sem dúvida a melhor opção de Unix para PC, pois possui todas as características um UNIX moderno, tais como: multitarefa real, multiusuário, memória virtual, biblioteca compartilhada, interface gráfica (X Windows) etc. O Linux possui centenas de comandos embutidos, chamado utilitários e ferramentas, cada ferramenta é um programa distinto, destinado a fazer um tarefa específica de forma rápida e segura, vide item 5.4.3 - Compisição dos diretórios do Linux.

O fato de ser um sistema aberto é extremamente flexível (usuário tem acesso ao fonte do sistema) é outro ponto positivo. O preço é outro atrativo US$ 44,00 (no Brasil, US$ 22,00 nos E.U.A), incluem 6 CD’s do Linux Developer’s Resourses CD-ROM, distribuido pela InfoMagic.

Diversos grupos de estudo do Linux no mundo inteiro garante atualizações do software praticamente mensais. Aliado a isto, cada nova versão incorpora dispositivos perífericos que são lançado no mercado, trazendo a seu usuário suporte as mais recentes conquista da indústria do hardware.

O Linux tem excelente mercado a nível acadêmico, o que nos faz crer constantes melhoras no software, pois grande parte dos melhores professores/pesquisadores de sistemas operacionais colaboram para o seu desenvolvimneto tecnológico.

A documentação é detalhada e completa (em inglês), em português é bastante escassa (ponto negativo). Toda a documentação pode ser facilmente acessada pela internet em diversos sites ou na documentação que acompanha o software.

O suporte técnico é um dos pontos fracos do sistema, é feito basicamente através da internet, não existe nenhuma empresa no Brasil especializada no suporte ao Linux.

O número de aplicativos é limitado e não existe a curto prazo pespectiva de entrada de grandes software house desenvolvendo aplicativos para Linux, sem aplicativos não há como um sistema se tornar popular no mundo dos PCs.

O sistema de arquivos varia de um distribuição a outro, tornado difícil a vida do administrador de sistema, que muitas vezes tem dificuldade de descobrir o que essencial em cada subdiretório, limpar, criar, preparar, verificar, encontar e montar outros sistemas de arquivos (possívelmente em máquinas remotas), todas estas tarefas podem ser dificultadas se não encontrarmos os arquivos aonde esperamos.

A administração e a operação de um modo geral é bem mais complexa em um ambiente unix, inclusive o Linux, do que no ambiente DOS/Windows, o que dificulta sua popularização.

A Interface Gráfica X-Windows, Xfree86 versão 3.2, ainda precisa ser melhorada, principalmente os aplicativos que são desenvolvidos para a mesma, ainda muito pobres comparada com a GUI do Windows95.

Conclusão

O Linux, é um sistema operacional do tipo Unix, o padrão System V esta embutido no seu kernel, foi desenvolvido para a plataforma IBM-PC, sendo assim, ele possui a robustez e segurança e flexibilidade do Unix. Além disso ele possui uma interface gráfica chamada de X Windows(XFree86 versão 3.2, por exemplo, existe outrasinterface gráfica), que é semelhante a do Windows 95, menos sofisticada, menos aplicativos e menos elaborada, porém funcional.

O Projeto Linux foi desenvolvido para ser uma arquitetura aberta, você terá toda a liberdade de desenvolver software para sua plataforma. Os fontes são distribuidos junto com o produto. Por outro lado, não existe um grande mercado para a plataforma Linux, sendo assim, as grandes softwares houses do mundo, como a Microsoft, Lotus, Corel, Borland, Novell, etc não se preocuparam em desenvolvem aplicativos para ele. O desenvolvimento de aplicativos ainda é pequeno, colaboradores, em sua maioria pesquisadores, desenvolve os softwares e a distribuição é feita, preferencialmente, no mesmo pacote a preços módicos, todo o pacote custa no Brasil US$ 44,00 (US$ 22,00 nos USA), pode ser comprado utilizando qualquer Cartão de Crédito Internacional.

Outro problema do Linux é a falta de suporte técnico, não existe a nível comercial no Brasil, as informações são obtidas através de diversos manuais contidos no software, ou através da internet. Existe uma farta documentação disponível na rede, diversos servidores WWW (World Wide Web), lista de discussões (serviços que permite o intercâmbio de mensagem entre vários usuários, funciona como uma extensão do correio eletrônico, onde qualquer mensagem enviada a este endereço fictício, conhecido como alias, e reenviada automaticamente para todos os endereços da lista associada), Netnews ou USENET ou NEWS (semelhante a lista de discussão, só que as mensagem são enviadas a um determinado computador da rede que as reenvia em bloco, para outros computadores que aceitam o serviço), FTP (File Transfer Protocol, serviço básico de tranferência de arquivos na rede), etc.

O mercado do Linux no Brasil é restrito, praticamente, ao meio acadêmico e alguns provedores da Internet (ex.: Universidade Federal de Goiás utiliza como roteador e servidor de WWW em algumas unidade acadêmicas). É difícil acreditar no crescimento do mercado Linux no Brasil a curto e a médio prazo fora deste nincho de mercado, devido a sua dificuldade de operação, o Linux foi projetado por programadores para programadores, a fim de ser utilizado em ambiente onde a maioria dos usuários tenha uma certa experiência, soma-se a isto a falta de software para a plataforma Linux que dificulta a sua popularização. As empresas que utilizam o Unix comercialmente na plataforma RISC, utilizam sistemas proprietários desenvolvidos por empresas como Sun, IBM, Dec, etc, que fazem tanto o hardware quanto SO, embora exista versões do Linux para algumas destas plataformas (ex. Sun) não acreditamos no crescimento do Linux neste mercado, pois é estritamente fechado.

Apesar disto, o Linux é uma opção séria como sistemas operacional do tipo Unix para o mundo PC, podemos recomendá-lo sem medo de errar, se sua empresa ou aplicação precisar de um sistema com as características do Unix, é você possue máquinas Intel ou compatíveis, pode utillizá-lo é a melhor opção hoje e com grandes possibilidades de crescimento nesta faixa de mercado.

Bibliografia

Servidores www

http://www.openline.com.br/Linux-br/ - Home Page do Linux no Brasil
http://www.br.freebsd.org/Linux-br/index.html - Home Page do Linux em português
http://www.inf.ufrgs.br/~kojima/Linux/faq-Linux.html - Perguntas frequentemente colocadas na Linux-br (FAQ)
http://www.Linux.org - Home page da Linux Organization, site oficial do Linux
http://www.suncite.unc.edu/mdw/welcome.html - Página do Projeto de Documentação do Linux.
http://www.Linux.if.usp.br - Tudo para Linux em português/inglês site da USP.
http://www.infor.es/LuCAS - Projeto Lucas - informações do Linux em espanhol.
http://www.cl.com.ac.uk/users/wj10/Linux-faq - Pergunta mais frequentes do Linux em inglês.
http://sunsite.unc.edu/mdw/Linux.html - Site com tudo sobre Linux em Inglês

Servidores de FTP

ftp.iis.com.br - diversos arquivos da internet.
ftp.versatec.com - contém diversos softwares para Linux.
ftp.ibp.fr : /pub/Linux (França)
ftp.cc.gatech.edu : /pub/Linux (EUA - sudeste: Suranet)
ftp.cdrom.com : /pub/Linux (EUA)
ftp.informatik.tu-muenchen.de : /pub/comp/os/Linux (Alemanha)
ftp.ibr.cs.tu-bs.de : /pub/os/Linux (Alemanha)
ftp.dfv.rwth-aachen.de : /pub/Linux (Alemanha)
ftp.informatik.rwth-aachen.de : /pub/Linux (Alemanha)
ftp.cc.monash.edu.au : /pub/Linux (Austrália)
ftp.dstc.edu.au : /pub/Linux (Austrália: Queensland)
ftp.sun.ac.za : /pub/Linux (África do Sul)
ftp.inf.utfsm.cl : /pub/Linux (Chile)
ftp.zel.fer.hr : /pub/Linux (Croácia)
Linux.if.usp.br : /pub/mirror/sunsite.unc.edu/Linux além de outros em /pub/mirror com a distribuicao Debian.
lcmi.ufsc.br : /pub/diversos/Linux (Brasil : Santa Catarina) Slackware
cesar.unicamp.br : /pub3/Linux (Brasil : São Paulo) Slackware
ftp.ime.usp.br : /pub/Linux (Brasil : São Paulo) Slackware
ftp.ufpr.br : /pub/Linux/ (Brasil : Paraná) Slackware

Lista de discussões (usernet newsgroup)

comp.os. Linux**.announce** - é um grupo de anúncios moderado; você deve lê-lo se pretende usar Linux. Submissões a este grupo devem ser mandadas para Linux-announce@news.ornl.gov.

comp.os. Linux**.answers** - Contém todos os FAQs, HOWTOs e outros documentos importantes. Assine este grupo também.

Os outros grupos na hierarquia comp.os.Linux.* também são recomendados algums problemas comuns não respondidos neste FAQ podem estar nos newsgroups. Esses grupos são :
comp.os. Linux**.setup**
comp.os. Linux**.hardware**
comp.os. Linux**.networking**

comp.os. Linux**.x**
comp.os. Linux**.development.apps,**
comp.os. Linux**.development.system**
comp.os. Linux**.advocacy**
comp.os. Linux**.misc.**

Endereço eletrônico de diversos colaboradores da Linux Organização

Drew Eckhard (US) drew@coorado .edu
Brondo n S. Allbere(US) bsa@kf8nh.wariat.org
Ian Jacksão (UK) ijacksão@cus.cam.ac.uk
Rik Faith (US) faith@cs.unc.edu
Iar Mc Coghrie (US) ian@ucsd.edu
Stephem Harris (UK) sweh@spudde .mew.co.uk
DaniOQuilan (US) Daniel.Quinlan@Linux.org
Fred N. vain Kemper(US) waltje@infomagic.com
Mike Sangree (US) mike@sojurn.lns.pa.us
Jhon A. Martir (US) jmartin@csc.com
David H. Selber (US) dhs@gowworm.firefle .com
Chris Netcalf(US) metcalf@lcs.mit.edu
Theodo re Tsó(US) te tso@athema.mit.edu
Ian Murdo ck (US) imurdo ck@debian.org
Stephem Tweedie (UK) sct@dcs.ed.ac.uk
David C. Niemi (US) niemid@clarck