De Quando é Ligado até o Bash

Greg O'Keefe, gcokeefe@postoffice.utas.edu.au v0.9a, Novembro 2000

Esta é uma breve descrição do que acontece num sistema Linux, de quando é ligado até o momento em que você se identifica e obtém uma linha de comandos do bash. Entender isso lhe será útil quando precisar resolver problemas ou configurar seu sistema.

Contents

1 Introdução

Acho frustrante que muitas coisas acontecem em minha máquina Linux que não entendo. Se você, como eu, também quer realmente entender seu sistema e não apenas saber como usá-lo, este documento é um bom começo. Esse tipo de conhecimento também é necessário se você quer se tornar um solucionador de problemas em Linux de primeira.

Presumo que você tenha uma máquina Linux funcionando, e entende algumas coisas básicas sobre Unix e hardware de PC. Se não, um excelente ponto de partida é o

The Unix and Internet Fundamentals HOWTO http://www.linuxdoc.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO.html de Eric S. Raymond. É curto, fácil de ler e trata de todos os temas básicos.

O principal tema deste documento é como o Linux se inicializa. Mas este também tenta ser um recurso de aprendizado mais abrangente. Eu inclui exercícios em cada seção. Se você fizer alguns destes, irá aprender muito mais que simplesmente lendo.

Espero que alguns leitores façam o melhor exercício para aprender Linux que conheço, que é construir um sistema a partir do código fonte. Giambattista Vico, um filósofo Italiano (1668-1744) disse "verum ipsum factum", que significa "o conhecimento vem pela prática". Obrigado ao Alex (ver 14.4 (Agradecimentos)) por esta citação.

Se você quer por a mão na massa, veja também o *Linux From Scratch HOWTO* http://www.linuxfromscratch.org (LFS) de Gerard Beekman. O LFS tem instruções detalhadas para construir um sistema usável completo a partir do código fonte. No sítio do LFS você também vai encontrar uma lista de discussões para pessoas que estão construindo sistemas dessa forma. As instruções que faziam parte deste documento agora estão em um documento separado, "Building a Minimal Linux System from Source Code", e pode ser encontrado em *From PowerUp to Bash Prompt home page* http://www.netspace.net.au/~gok/power2bash/ . Elas indicam como construir um sistema experimental, como um mero exercício de aprendizagem.

Os pacotes são apresentados na ordem em que eles aparecem no processo de inicialização do sistema. Isso significa que se você instalar os pacotes nesta ordem poderá reiniciar após cada instalação e ver o sistema ir tomando forma, até te fornecer a linha de comandos do bash. Há uma forte sensação de progresso.

Recomendo primeiro ler o texto principal de cada seção, pulando os exercícios e referências. Decida então até onde deseja se aprofundar e quanto de esforço você está disposto a investir. Em seguida volte ao início, fazendo os exercícios e as leituras adicionais conforme for avançando.

2 Hardware

Quando você liga o computador, ele faz um teste em si mesmo para certificar-se de que tudo está funcionando corretamente. Esse teste é chamado de "Power on self test". Em seguida um programa chamado bootstrap loader, que se encontra na ROM BIOS, procura por um setor de inicialização. Um setor de inicialização é o primeiro setor em um disco e tem um pequeno programa capaz de carregar um sistema operacional. Os setores de inicialização são marcados com um número mágico 0xAA55 = 43603 no byte 0x1FE = 510. Estes são os dois últimos bytes do setor. É desta forma que o hardware pode saber se um setor é um setor de inicialização ou não.

O bootstrap loader tem uma lista de lugares onde procurar por um setor de inicialização. Minha máquina velha procura no drive primário de disquete, em seguida no disco rígido primário. Máquinas mais modernas também podem procurar por um setor de inicialização num CD-ROM. Se encontrar um setor de inicialização, irá carregá-lo na memória e passar o controle ao programa que carrega o sistema operacional. Num sistema Linux típico este programa será o primeiro estágio do carregador LILO. Existem muitas maneiras diferentes de configurar a inicialização do seu sistema. Veja o *LILO User's Guide* para mais detalhes. Na seção 3.3 (LILO) há uma URL.

Obviamente há muito mais pra dizer sobre o que o hardware dum PC faz, mas este não é o lugar para isto. Veja um dos muitos bons livros sobre hardware de PC.

2.1 Configuração

A máquina armazena alguma informação sobre si mesma em sua CMOS. Isto inclui quais discos e RAM existem no sistema. A BIOS da máquina contém um programa que permite a você modificar estas configurações. Veja a mensagem na tela de sua máquina quando é ligada para verificar como acessá-la. Em minha máquina se pressiona a tecla delete antes que ela comece a carregar o sistema operacional.

2.2 Exercícios

Um bom jeito de aprender sobre o hardware do PC é montar uma máquina com componentes de segunda mão. Use pelo menos um 386, assim você poderá colocar o Linux para rodar facilmente. Não será muito caro. Pergunte por aí, alguém poderia te dar algumas peças que vais precisar.

Baixe, compile e faça um disco de inicialização para o *Unios* `http://www.netspace.net.au/~gok/resources` . (Eles tinham uma página em `http://www.unios.org` mas desapareceu). É um simples programa "Olá mundo!", que consiste em pouco mais de 100 linhas de código assembly. Seria bom vê-lo convertido a um formato que possa ser entendido pelo montador GNU `as`.

Existem instruções para criar seu próprio sistema operacional em *Roll Your Own* `http://www.acm.uiuc.edu/sigops/roll_your_own/` se você quiser um desafio *de verdade*.

Abra a imagem de disco do unios com um editor hexadecimal. Esta imagem tem 512 bytes, exatamente um setor. Encontre o número mágico 0xAA55. Faça o mesmo para o setor de inicialização dum disquete em seu próprio computador. Você pode usar o comando `dd` para copiar ele para um arquivo: `dd if=/dev/fd0 of=boot.sector`. Seja *muito* cuidadoso escrevendo `if` (input file) e `of` (output file) corretamente!

Veja o código fonte do carregador de sistemas LILO.

2.3 Mais Informações

- *The Unix and Internet Fundamentals HOWTO* `http://www.linuxdoc.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO.html` por Eric S. Raymond, especialmente a seção 3, *What happens when you switch on a computer?*
- O primeiro capítulo do *The LILO User's Guide* tem uma excelente explicação sobre as partições do disco e a inicialização do PC. Na seção 3.3 (LILO) há uma URL.
- *The NEW Peter Norton Programmer's Guide to the IBM PC & PS/2*, por Peter Norton e Richard Wilton, Microsoft Press 1988. Há um livro novo do Norton, parece bom, mas agora eu não posso pagar.
- Um dos muitos livros disponíveis sobre atualização de PCs.

3 Lilo

Quando o computador carrega um setor de inicialização em um sistema Linux normal, o que ele realmente carrega é uma parte do lilo chamado "first stage boot loader" (primeiro estágio do carregador). É um pequeno programa que a única coisa que faz na vida é carregar e rodar o segundo estágio.

O segundo estágio apresenta uma linha de comandos (se foi instalado assim) e carrega o sistema operacional que você escolher.

Quando seu sistema está em funcionamento e você roda `lilo`, o que você está realmente rodando é um "instalador do mapa". Ele lê o arquivo de configuração `/etc/lilo.conf` e escreve os carregadores, assim como informações sobre os sistemas operacionais que ele pode carregar, no disco rígido.

Existem diversas formas de configurar a inicialização de seu sistema. O que acabo de explicar é a forma mais óbvia e "normal", pelo menos em um sistema onde o principal sistema operacional é o Linux. O Guia do Usuário do LILO explica vários exemplos de conceitos de inicialização. Vale a pena ler e tentar alguns deles.

3.1 Configuração

O arquivo de configuração para o lilo é `/etc/lilo.conf`. Há uma página de manual para ele: digite `man lilo.conf` num shell para vê-la. O principal no `lilo.conf` é uma entrada para cada sistema operacional que o lilo pode inicializar. Em uma entrada para Linux inclui onde está o kernel e que partição do disco montar como raíz. Para outros sistemas operacionais, a informação mais importante é a partir de qual partição inicializar.

3.2 Exercícios

PERIGO: tome cuidado com estes exercícios. É bastante fácil fazer algo errado e danificar seu MBR (master boot record - registro mestre de inicialização) e tornar seu sistema inutilizável. Certifique-se de ter um disco de recuperação funcionando e saber como usá-lo para concertar a bagunça. Abaixo há um link para o tomsrtbt, o disco de recuperação que eu uso e recomendo. A melhor precaução é usar uma máquina sem importância.

Prepare um disquete com o lilo. Não importa que não há nada além do kernel no disquete - você verá um "kernel panic" quando o kernel estiver pronto para carregar o init, mas pelo menos você saberá que o lilo está funcionando.

Se quiser, você pode espremer para ver quanto do sistema você consegue colocar num disquete. Esta é provavelmente a segunda melhor atividade para aprender Linux. Veja o Bootdisk HOWTO (url abaixo) e tomsrtbt (url abaixo) para ter uma idéia.

Faça o lilo inicializar o unios (há uma URL na seção 2.2 (exercícios de hardware)). Como um desafio extra tente fazer isto num disquete.

Faça um laço de inicializações. Configure o lilo no MBR para inicializar o lilo no setor de inicialização de uma partição primária, e este para inicializar o lilo no MBR... Ou talvez usar o MBR e todas as quatro partições primárias, fazendo um laço de cinco pontos. Divirta-se!

3.3 Mais Informações

- A página de manual do lilo.
- O pacote Lilo (*lilo* `ftp://lrcftp.epfl.ch/pub/linux/local/lilo/`), contém o "LILO User's Guide"`lilo-u-21.ps.gz` (ou outra versão mais recente). É provável que você já tenha este documento. Tente em `/usr/doc/lilo` ou algo parecido. A versão postscript é melhor que a de texto simples, já que contém diagramas e tabelas.
- *tomsrtbt* `http://www.toms.net/rb` o melhor linux em um só disquete. Excelente como disco de recuperação.
- *The Bootdisk HOWTO* `http://www.linuxdoc.org/HOWTO/Bootdisk-HOWTO/`

4 O Kernel do Linux

Na verdade o kernel faz muitas coisas. Creio que uma boa forma de resumir é que ele faz o hardware fazer o que os programas querem, de forma clara e eficiente.

O processador pode executar apenas uma instrução de cada vez, mas os sistemas Linux parecem fazer várias coisas simultaneamente. O kernel consegue isto alternando de uma tarefa à outra rapidamente. Ele faz um melhor uso do processador rastreando quais processos estão prontos e quais estão esperando por algo como um registro num arquivo do disco rígido ou uma entrada do teclado. Esta tarefa do kernel é chamada de scheduling (agendamento).

Se um programa não está fazendo nada, não precisa estar na RAM. Mesmo um programa que está fazendo algo pode ter partes que não estão fazendo nada. O espaço de endereçamento de cada processo está dividido em páginas. O kernel verifica quais páginas estão sendo usadas mais freqüentemente. As páginas que não estão sendo muito usadas podem ser retiradas para a partição de troca (swap). Quando elas forem necessárias novamente, pode se retirar outra página pouco usada para desocupar-lhe espaço. Este é o gerenciamento de memória virtual.

Se alguma vez você já compilou seu próprio Kernel, você deve ter notado que existem muitas opções para dispositivos específicos. O kernel contém muito código específico para comunicar com os diversos tipos de hardware, e apresentá-los de uma maneira uniforme aos aplicativos.

O Kernel também gerencia o sistema de arquivos, a comunicação entre os processos, e um monte de coisas relacionadas à redes.

Uma vez que o kernel for carregado, a primeira coisa que ele faz é procurar por um programa `init` para executar.

4.1 Configuração

A maior parte da configuração do kernel é feita quando o estiver construindo, usando `make menuconfig`, ou `make xconfig` em `/usr/src/linux/` (ou onde quer que esteja o código fonte do kernel do seu Linux). Você pode redefinir o modo de vídeo padrão, o sistema de arquivos raíz, o dispositivo de troca (swap) e o tamanho do disco em RAM usando `rdev`. Estes e outros parâmetros podem ser passados ao kernel pelo lilo. Você pode dar ao lilo os parâmetros para passar ao kernel tanto no lilo.conf quanto na linha de comandos do lilo. Por exemplo, se quiser usar hda3 como seu sistema de arquivos raíz em vez de hda2, poderia digitar:

```
LILO: linux root=/dev/hda3
```

Se estiver construindo um sistema a partir do código fonte, será bem mais fácil criar um kernel "monolítico", ou seja, sem módulos. Assim não irá precisar copiar os módulos para o sistema de destino.

NOTA: O agente de log do kernel usa o arquivo `System.map` para determinar os nomes dos módulos gerando mensagens. O programa `top` também usa esta informação. Quando copiar o kernel para o sistema de destino, copie também `System.map`.

4.2 Exercícios

Pense nisto: `/dev/hda3` é um tipo especial de arquivo que descreve uma partição do disco rígido. Mas ele está num sistema de arquivos exatamente como todos os outros arquivos. O kernel quer saber qual partição montar como sistema de arquivos raíz - ele ainda não tem um sistema de arquivos. Então como ele pode ler `/dev/hda3` para saber qual partição montar?

Se ainda não o fez: construa seu próprio kernel. Leia toda a informação de ajuda para cada opção.

Tente fazer o menor kernel funcional possível. Você pode aprender bastante quando as coisas derem errado!

Leia "The Linux Kernel" (URL abaixo) e veja se pode encontrar as partes do código fonte a qual ele se refere. O livro (no momento em que escrevo) usa a versão 2.0.33 do kernel, que está obsoleta. Será mais fácil se baixar esta versão antiga e ler estes fontes. É surpreendente encontrar coisas no código C chamados "process" e "page".

Fuce! Veja se consegue fazer ele mostrar umas mensagens extras ou qualquer outra coisa.

4.3 Mais Informações

- `/usr/src/linux/README` e o conteúdo de `/usr/src/linux/Documentation/` (eles podem estar em algum outro lugar em seu sistema)
- *The Kernel HOWTO* `http://mirror.aarnet.edu.au/linux/LDP/HOWTO/Kernel-HOWTO.html`
- A ajuda disponível ao configurar um kernel usando `make menuconfig` ou `make xconfig`
- *The Linux Kernel (e outros guias LDP)* `http://mirror.aarnet.edu.au/linux/LDP/LDP/`
- código fonte, veja *Building a Minimal Linux System from Source Code* `http://www.netspace.net.au/~gok/power2bash` para encontrar urls

5 A Biblioteca GNU C

A próxima coisa que acontece quando seu computador é inicializado é carregar e rodar o init. Porém, o init, como quase todos programas, usa funções de bibliotecas.

Você já deve ter visto um programa de exemplo em C como este:

```
main() {
        printf("Hello World!\n");
}
```

O programa não contém a definição de `printf`, então de onde ela vem? Ela vem das bibliotecas C padrão, num sistema GNU/Linux, glibc. Se for compilado com o Visual C++, então será a implementação da Microsoft das mesmas funções padrão. Existem zilhões destas funções padrão, para matemática, string, data e hora, alocação de memória, e assim por diante. Tudo no Unix (incluindo Linux) é escrito em C ou se esforça em fingir que o é, portanto tudo usa estas funções.

Se você olhar em `/lib` em seu sistema linux, verá arquivos chamados `libqualquercoisa.so` ou `libqualquercoisa.a` aos montes. São as bibliotecas destas funções. Glibc é apenas a implementação GNU destas funções.

Existem duas formas pelas quais os programas podem usar estas funções de bibliotecas. Se vincular *estaticamente*, estas funções de bibliotecas serão copiadas no executável que será criado. Para isso que servem as bibliotecas `libqualquercoisa.a`. Se vincular *dinamicamente* (e este é o padrão), então quando estiver rodando e precisar do código da biblioteca, ele será chamado a partir do arquivo `libqualquercoisa.so`.

O comando `ldd` é seu amigo quando quiser saber quais bibliotecas são necessárias à um programa em particular. Por exemplo, aqui estão as bibliotecas que o `bash` usa:

```
[greg@Curry power2bash]$ ldd /bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40019000)
        libc.so.6 => /lib/libc.so.6 (0x4001d000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

5.1 Configuração

Algumas das funções das bibliotecas dependem de onde você está. Por exemplo, na Austrália nós escrevemos as datas como dd/mm/aa, mas os Americanos escrevem mm/dd/aa. Existe um programa que acompanha a distribuição da `glibc` chamado `localedef` que te permite definir isso.

5.2 Exercícios

Use o `ldd` para verificar quais bibliotecas seus aplicativos favoritos usam.

Use o `ldd` para verificar quais bibliotecas o `init` usa.

Faça uma biblioteca de testes, apenas com uma ou duas funções. O programa `ar` é usado para criá-las, a página de manual para o `ar` pode ser um bom lugar para começar a investigar como isto é feito. Escreva, compile e vincule um programa que use esta biblioteca.

5.3 Mais Informações

- código fonte, veja *Building a Minimal Linux System from Source Code* `http://www.netspace.net.au/~gok/power2bash` para encontrar urls

6 Init

Vou falar apenas sobre o init estilo "System V" que é majoritário nos sistemas Linux. Existem alternativas. De fato, você pode pôr qualquer programa que queira em `/sbin/init`, e o kernel vai rodá-lo assim que terminar de carregar.

É tarefa do `init` pôr tudo para funcionar de forma apropriada. Ele verifica se os sistemas de arquivos estão corretos e os montam. Inicializa os "daemons" para registrar as mensagens do sistema, redes, servir páginas web, escutar o mouse e tudo mais. Também inicializa os processos "getty" que colocam as linhas de comandos (prompt) de login em seus terminais virtuais.

Há um contexto muito complicado que aborda a alternância entre níveis de execução (run-levels), mas vou pular a maior parte disso e vou falar apenas sobre a inicialização do sistema.

O init lê o arquivo `/etc/inittab`, que lhe diz o que fazer. Tipicamente, a primeira coisa que lhe é instruída a fazer é rodar o roteiro (script) de inicialização. O programa que executa (ou interpreta) este roteiro é o `bash`, o mesmo programa que te fornece a linha de comandos. Em sistemas Debian, o roteiro de inicialização é `/etc/init.d/rcS`, no Red Hat é `/etc/rc.d/rc.sysinit`. Este é o local onde os sistemas de arquivos são checados e montados, o relógio é ajustado, os espaços de troca (swap) habilitados, o nome da máquina é definido, etc.

Depois chama outro roteiro que nos dá o nível de execução padrão, que é apenas inicializar um conjunto de subsistemas. Existe um conjunto de diretórios `/etc/rc.d/rc0.d`, `/etc/rc.d/rc1.d`, ..., `/etc/rc.d/rc6.d` no Red Hat, ou `/etc/rc0.d`, `/etc/rc1.d`, ..., `/etc/rc6.d` no Debian, que correspondem aos níveis de execução. Se formos para o nível de execução 3 num sistema Debian, então o roteiro roda todos os roteiros em `/etc/rc3.d` que começam com 'S' (de start). Estes scripts são apenas vínculos para os roteiros em outro diretório, normalmente chamado `init.d`.

Portanto nosso roteiro de nível de execução é chamado `init`, e está procurando num diretório por roteiros começando com 'S'. Talvez encontre primeiro `S10syslog`. Os números dizem ao roteiro de nível de execução qual a ordem em que devem ser executados. Neste caso `S10syslog` é o primeiro, porque não existem roteiros começando com S00 .. S09. Porém `S10syslog` é na verdade um vínculo para `/etc/init.d/syslog`, que é o roteiro para iniciar e parar o agente de log do sistema. O roteiro de nível de execução sabe que deve executar o roteiro `syslog` com o parâmetro "start" porque o vínculo começa com 'S'. Existem os vínculos correspondentes começando com 'K' (de kill), que especificam o que terminar e em que ordem quando estivermos saindo do nível de execução.

Para modificar quais subsistemas iniciar por padrão, defina os vínculos no diretório `rcN.d`, onde N é o nível de execução padrão em seu `inittab`.

A última coisa importante que o init faz é iniciar alguns `getty`. Se algum deles parar, é "recriado" pelo `init`. A maioria das distribuições vem com seis terminais virtuais. Você pode querer menos para economizar memória, ou mais para ter muitas coisas rodando ao mesmo tempo e visualizá-las quando for necessário rapidamente. Você pode querer também executar um `getty` para um terminal modo texto ou uma discagem em modem. Nesse caso você devera editar o arquivo `inittab`.

6.1 Configuração

`/etc/inittab` é o arquivo de configuração de mais alto nível para o init.

Os diretórios `rcN.d`, onde N = 0, 1, ..., 6 determinam quais subsistemas serão inicializados.

Em algum lugar num dos roteiros invocados pelo init, aparecerá o comando `mount -a`. Isto significa montar todos os sistemas arquivos que devem ser montados. O arquivo `/etc/fstab` define o que deve ser montado. Se quiser mudar o que será montado quando seu sistema inicializar, é este arquivo que você deve editar. Há uma página de manual para o `fstab`.

6.2 Exercícios

Encontre o diretório `rcN.d` para o nível de execução padrão de seu sistema e digite `ls -l` para ver para quais arquivos os vínculos apontam.

Modifique o número de gettys que o seu sistema roda.

Remova qualquer subsistema desnecessário do seu nível de execução padrão.

Veja o quanto você pode reduzir a inicialização.

Prepare um disquete com o lilo, um kernel e um programa "olá mundo" vinculado estaticamente chamado `/sbin/init` e veja ele inicializar e dizer olá.

Observe atenciosamente a inicialização de seu sistema e tome notas sobre o que ele diz estar acontecendo. Ou imprima uma seção do log do seu sistema `/var/log/messages` desde o momento da inicialização. Então começando pelo `inittab`, siga todos os roteiros e veja o que faz cada código. Você também pode colocar mensagens de inicialização extras nele, como por exemplo:

```
echo "Olá, sou o rc.sysinit"
```

Este também é um bom exercício para aprender escrever roteiros de Bash, alguns dos roteiros são bastante complicados. Tenha um bom guia de referência do Bash a mão.

6.3 Mais Informações

- Existem páginas de manual para os arquivos `inittab` e `fstab`. Digite (por exemplo) `man inittab` num shell para vê-las.
- The Linux System Administrators Guide tem uma boa *seção* http://mirror.aarnet.edu.au/linux/LDP/LDP/ sobre o init.

- código fonte, veja

  Building a Minimal Linux System from Source Code http://www.netspace.net.au/~gok/power2bash para encontrar urls

7 O sistema de arquivos

Nesta seção, estarei usando a expressão "sistema de arquivos" de duas formas diferentes. Existem sistemas de arquivos em partições do disco ou outros dispositivos, e existe o sistema de arquivos como lhe é apresentado pelo sistema Linux em funcionamento. No Linux, você "monta" um sistema de arquivos em disco no sistema de arquivos do sistema.

Na seção anterior mencionei que os roteiros do init verificam e montam sistemas de arquivos. Os comandos que fazem isso são `fsck` e `mount` respectivamente.

Um disco rígido é apenas um grande espaço onde você pode escrever zeros e uns. Um sistema de arquivos impõe certa estrutura nisto, fazendo parecer que existem arquivos em seus diretórios e estes em seus diretórios... Cada arquivo é representado por um inode, que diz a quem pertence o arquivo, quando foi criado e onde encontrar seu conteúdo. Diretórios também são representados por inodes, mas estes dizem onde encontrar os inodes dos arquivos que estão no diretório. Se o sistema quer ler `/home/greg/bigboobs.jpeg`, primeiro ele encontra o inode para o diretório raíz `/` no "superbloco", em seguida encontra o inode para o diretório `home` no conteúdo de `/`, depois encontra o inode para o diretório `greg` no conteúdo de `/home`, finalmente o inode para `bigboobs.jpeg` que irá dizer quais blocos do disco ler.

Se acrescentarmos algum dado ao fim do arquivo, pode acontecer dos dados serem escritos antes do inode ser atualizado para dizer quais os novos blocos pertencem ao arquivo, ou vice versa. Se houver um corte de energia neste ponto, o sistema de arquivos estará corrompido. É este tipo de coisa que o `fsck` tenta detectar e reparar.

O comando mount pega um sistema de arquivos num dispositivo, e o acrescenta à hierarquia que você vê quando usa seu sistema. Normalmente, o kernel monta o sistema de arquivos raíz somente para leitura. O comando mount é usado para remontá-lo para leitura e escrita depois que o `fsck` verificou que está tudo bem.

O Linux também suporta outros tipos de sistema de arquivos: msdos, vfat, minix e outros mais. Os detalhes de um tipo específico de sistema de arquivos são abstraídos pelo sistema de arquivos virtual (VFS). Não vou entrar em detalhes sobre isto. Existe uma discussão sobre isso no "The Linux Kernel" (há uma URL na seção 4.3 (O Kernel do Linux)).

Um tipo de sistema de arquivos completamente diferente é montado em `/proc`. É na verdade uma representação das coisas no kernel. Há um diretório ali para cada processo rodando no sistema, onde o número do processo é o nome do diretório. Também existem arquivos como `interrupts` e `meminfo` que dizem sobre como o hardware está sendo usado. Você pode aprender bastante ao explorar `/proc`.

7.1 Configuração

Existem parâmetros para o comando `mke2fs` que cria sistemas de arquivos ext2. Estes controlam o tamanho dos blocos, o número de inodes, e assim por diante. Veja a página de manual do `mke2fs` para mais detalhes.

O que é montado em seu sistema de arquivos é controlado pelo arquivo `/etc/fstab`. Também há uma página de manual para ele.

7.2 Exercícios

Faça um sistema de arquivos bem pequeno, e visualize-o com um visualizador hexadecimal. Identifique o superbloco, os inodes e o conteúdo dos arquivos.

Acredito que existam ferramentas que te dão uma visão gráfica do sistema de arquivos. Encontre uma, teste, e me mande um e-mail com a url e comentários.

Veja o código do kernel para o sistema de arquivos ext2.

7.3 Mais Informações

- O capítulo 9 do livro LDP "The Linux Kernel" tem uma excelente descrição dos sistemas de arquivos. Você pode encontrá-lo no *espelho* `http://mirror.aarnet.edu.au/linux/LDP/LDP/` Australiano do LDP.
- O comando `mount` é parte do pacote util-linux; há uma URL em

  Building a Minimal Linux System from Source Code `http://www.netspace.net.au/~gok/power2bash`
- Páginas de manual para o `mount`, `fstab`, `fsck`, `mke2fs` e `proc`.
- O arquivo `Documentation/proc.txt` no código fonte do Linux explica o sistema de arquivos `/proc`.
- A página do EXT2 File System Utilities *ext2fsprogs* `http://web.mit.edu/tytso/www/linux/e2fsprogs.html` e o *espelho* `ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/filesystems/ext2/` Australiano. Há ali um documento Ext2fs-overview, porém está obsoleto e não é tão legível quanto o capítulo nove do "The Linux Kernel".
- *Unix File System Standard* `ftp://tsx-11.mit.edu/pub/linux/docs/linux-standards/fsstnd/` (outro *link* `http://www.pathname.com/fhs/`). Este descreve o que deve ter em um sistema de arquivos Unix e por que. Também os requisitos mínimos para `/bin`, `/sbin` e outros. Esta é uma boa referência se seu objetivo é criar um sistema mínimo porém completo.

8 Daemons do Kernel

Ao executar o comando `ps aux`, verá algo semelhante a isto:

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
root         1  0.1  8.0  1284   536   ? S    07:37   0:04 init [2]
root         2  0.0  0.0     0     0   ? SW   07:37   0:00 (kflushd)
root         3  0.0  0.0     0     0   ? SW   07:37   0:00 (kupdate)
root         4  0.0  0.0     0     0   ? SW   07:37   0:00 (kpiod)
root         5  0.0  0.0     0     0   ? SW   07:37   0:00 (kswapd)
root        52  0.0 10.7  1552   716   ? S    07:38   0:01 syslogd -m 0
root        54  0.0  7.1  1276   480   ? S    07:38   0:00 klogd
root        56  0.3 17.3  2232  1156   1 S    07:38   0:13 -bash
root        57  0.0  7.1  1272   480   2 S    07:38   0:01 /sbin/agetty 38400 tt
root        64  0.1  7.2  1272   484  S1 S    08:16   0:01 /sbin/agetty -L ttyS1
root        70  0.0 10.6  1472   708   1 R   Sep 11   0:01 ps aux
```

Esta é a lista de processos rodando no sistema. A informação é dada pelo sistema de arquivos `/proc`, que mencionei na seção anterior. Note que o `init` é o processo número um. Os processos dois, três, quatro e cinco são kflushd, kupdate, kpiod e kswapd. Contudo há algo estranho aqui: note que tanto na coluna do tamanho do armazenamento virtual (SIZE) quanto o tamanho do armazenamento real (Real Storage Size, RSS) estes processos tem zeros. Como pode um processo não usar memória?

Estes processos são daemons do kernel. A maior parte do kernel não é mostrada na lista de processos, e você apenas poderá calcular quanta memória ele está usando subtraindo a memória disponível da que há em seu sistema. Os daemons do kernel são iniciados após o init, portanto recebem PID's (números de processo) assim como processos normais. Porém, seu código e dados residem na parte de memória do kernel.

A coluna de comando está entre parênteses porque o sistema de arquivos `/proc` não tem a informação da linha de comandos para estes processos.

Então para que servem estes daemons do kernel? Versões anteriores deste documento tinha um pedido de ajuda, já que eu não sei muito sobre os daemons do kernel. A história parcial que segue foi feita com pedaços das várias respostas deste pedido, pelas quais eu sou muito grato. Mais dicas, referências e correções são bem-vindas!

A entrada e a saída é feita através de *buffers* na memória. Isto permite as coisas rodarem mais rápido. O que os programas escrevem pode ser mantido na memória, num buffer, depois escrito para o disco em pedaços maiores e mais eficientes. Os daemons `kflushd` e `kupdate` controlam esta tarefa: `kupdate` roda periodicamente (5 segundos?) para verificar se existe algum buffer cheio. Se existir, faz com que `kflushd` o escreva para o disco.

Muitas vezes os processos não tem nada para fazer, e os que estão rodando às vezes não precisam ter todo o seu código na memória. Isto significa que podemos fazer melhor uso da memória retirando as partes não usadas dos programas em execução para a(s) partição(s) de troca (swap) do disco rígido. Mover estes dados para dentro e para fora da memória quando necessário é feito pelo `kpiod` e `kswapd`. Mais ou menos a cada segundo, `kswapd` acorda para verificar a situação da memória, e se alguma coisa no disco é necessária na memória ou não há memória livre suficiente, `kpiod` é chamado.

Também pode haver o daemon `kapmd` rodando em seu sistema se você configurou o gerenciamento automático de energia em seu kernel.

8.1 Configuração

O programa `update` te permite configurar o `kflushd` e `kswapd`. Experimente `update -h` para mais informações.

O espaço de troca é ativado por `swapon` e desativado por `swapoff`. O roteiro do init (`/etc/rc.sysinit` ou `/etc/rc.d/rc.sysinit`) normalmente chama `swapon` quando o sistema inicializa. Me disseram que `swapoff` é útil para economizar energia nos portáteis.

8.2 Exercícios

Execute `update -d`, note a frase na última linha sobre "threshold for buffer fratricide" (limite para o fratricídio do buffer). Aqui está um conceito intrigante, investigue!

Vá ao diretório `/proc/sys/vm` e visualize os arquivos ali com o `cat`. Veja se consegue entender.

8.3 Mais Informações

"The Linux Kernel" de The Linux Documentation Project (veja a seção 4.3 (O Kernel do Linux) para encontrar uma url)

O código fonte do kernel do Linux, se você for corajoso suficiente. O código do `kswapd` está em `linux/mm/vmscan.c`, e `kflushd` e `kupdate` estão em `linux/fs/buffer.c`.

9 O agente de log do sistema

O init inicializa os daemons `syslogd` e o `klogd`. Eles escrevem as mensagens para logs. As mensagens do kernel são manipuladas pelo `klogd`, enquanto o `syslogd` manipula as mensagens de outros processos. O registro de log principal é `/var/log/messages`. Quando alguma coisa está errada, este é um bom lugar para procurar o que é. Muitas vezes existem dicas valiosas ali.

9.1 Configuração

O arquivo `/etc/syslog.conf` diz aos agentes de log quais mensagens registrar e onde. As mensagens são identificadas pelos serviços de onde se originam e qual nível de prioridade elas têm. Este arquivo de configuração consiste de linhas que dizem que as mensagens do serviço x com prioridade y vão para z, onde z é um arquivo, tty, impressora, uma máquina remota, ou qualquer outra coisa.

NOTA: O syslog exige que o arquivo `/etc/services` esteja presente. O arquivo services aloca portas. Não estou certo se o syslog precisa de uma porta para poder fazer o registro remotamente, ou se mesmo o registro local é feito através uma porta, ou se ele usa o arquivo `/etc/services` apenas para converter os nomes dos serviços que você digita no arquivo `/etc/syslog.conf` em números de portas.

9.2 Exercícios

Dê uma olhada no log do seu sistema. Encontre uma mensagem que você não compreende e descubra o que significa.

Mande todas as mensagens do seu log para um tty. (depois coloque tudo de volta ao normal)

9.3 Mais Informações

Espelho http://mirror.aarnet.edu.au/pub/linux/metalab/system/daemons/ Australiano do sysklogd

10 Getty e Login

Getty é o programa que te permite efetuar login através de um dispositivo serial como um terminal virtual, um terminal texto, ou um modem. Ele mostra a linha de comandos do login. Uma vez que você introduz seu nome de usuário, o getty o entrega para o `login`, que te pede a senha, a verifica e te fornece um shell.

Existem muitos gettys disponíveis. Algumas distribuições, incluindo a Red Hat, usam um muito pequeno chamado `mingetty` que funciona apenas com terminais virtuais.

O programa `login` é parte do pacote util-linux, que também contém um getty chamado `agetty`, que funciona muito bem. Este pacote também contém `mkswap`, `fdisk`, `passwd`, `kill`, `setterm`, `mount`, `swapon`, `rdev`, `renice`, `more` (o programa) e mais (quer dizer, mais programas).

10.1 Configuração

A mensagem que aparece no topo da sua tela com a linha de comandos do login vem de `/etc/issue`. Os gettys geralmente são iniciados por `/etc/inittab`. O login verifica os detalhes sobre o usuário em `/etc/passwd` e, se o sistema tem mascaramento de senhas, `/etc/shadow`.

10.2 Exercícios

Crie um `/etc/passwd` manualmente. As senhas podem ser nulas, e mudadas com o programa `passwd` uma vez que você efetuar login. Veja a página de manual para este arquivo, use `man 5 passwd` para obter a página de manual para o arquivo ao invés da página de manual para o programa.

11 Bash

Se você der ao `login` uma combinação válida do nome de usuário e senha, ele irá verificar em `/etc/passwd` qual shell te fornecer. Na maioria dos casos num sistema Linux será o `bash`. É tarefa do `bash` ler seus comandos e fazer com que sejam executados. Ele é simultaneamente uma interface de usuário e um interpretador de uma linguagem de programação.

Como interface de usuário ele lê seus comandos e os executa se forem comandos "internos" como `cd`, ou procura e executa um programa se ele for "externo" como `cp` ou `startx`. Ele também faz coisas excelentes como manter um histórico dos comandos e o completamento dos nomes de arquivos.

Nós já vimos o `bash` em ação como interpretador de linguagem de programação. Os roteiros que o `init` roda para inicializar o sistema normalmente são roteiros shell, e são executados pelo `bash`. Ter uma verdadeira linguagem de programação junto com os utilitários habituais do sistema disponíveis na linha de comandos é uma combinação muito poderosa, se souber o que estiver fazendo. Por exemplo (modo convencido ativado) outro dia precisei aplicar um monte de "remendos" a um diretório de código fonte. Fui capaz de fazê-lo usando apenas este comando:

```
for f in /home/greg/sh-utils-1.16*.patch; do patch -p0 < $f; done;
```

Este comando procura em todos os arquivos em meu diretório pessoal cujos nomes começam com `sh-utils-1.16` e terminam com `.patch`. Para cada arquivo encontrado ele aponta a variável `f` para aquele arquivo e executa os comandos entre `do` e `done`. Neste caso havia 11 arquivos para "remendar", mas seria igualmente fácil se houvesse três mil.

11.1 Configuração

O arquivo `/etc/profile` controla o comportamento do bash para todo o sistema. O que você colocar aqui irá afetar todo mundo que usa o bash em seu sistema. Ele fará coisas como adicionar diretórios ao `PATH`, definir sua variável de diretório `MAIL`, etc.

O comportamento padrão do teclado muitas vezes deixa muito a desejar. Na verdade é o readline que manipula isto. O readline é um pacote separado que manipula interfaces de linha de comandos, oferecendo o histórico dos comando e o completamento de nomes de arquivos, assim como algumas características avançadas de edição de linhas. Está compilado no bash. Por padrão, o readline é configurado usando o arquivo `.inputrc` em seu diretório pessoal. A variável do bash INPUTRC pode ser usada para substituir isto para o bash. Por exemplo: no Red Hat 6, `INPUTRC` é definida em `/etc/profile` para `/etc/inputrc`. Isto significa que as teclas de retrocesso (backspace), apagar (delete), início (home) e fim (end) funcionam bem para todo mundo.

Uma vez que o bash leu o arquivo de configuração para todo o sistema, ele procura pelo seu arquivo de configuração pessoal. Ele procura em seu diretório pessoal por `.bash_profile`, `.bash_login` e `.profile`. Ele roda o primeiro que encontrar. Se quiser mudar a forma que o bash se comporta para você, sem modificar o modo que se comporta para os outros, faça isto aqui. Por exemplo, muitos aplicativos usam variáveis de ambiente para controlar seu funcionamento. Eu tenho a variável `EDITOR` definida para `vi` de forma que posso usar o vi no Midnight Commander (um excelente gerenciador de arquivos no modo console) em vez do seu próprio editor.

11.2 Exercícios

O básico do bash é fácil de aprender. Mas não pare por aí: há uma incrível profundidade nele. Habitue-se procurar meios melhores de se fazer as coisas.

Leia os roteiros shell, procure saber o que não entende.

11.3 Mais Informações

- Existe um "Bash Reference Manual" que é abrangente, porém de difícil leitura.
- Existe um livro da O'Rielly sobre o Bash, mas não estou certo se é bom.
- Não conheço nenhum bom tutorial gratuito e atualizado sobre o bash. Se souber, por favor me envie por e-mail uma url.
- código fonte, veja *Building a Minimal Linux System from Source Code* `http://www.netspace.net.au/~gok/power2bash` para encontrar urls.

12 Comandos

No bash a maioria das coisas é feita por meio de comandos como `cp`. Quase todos estes comandos são pequenos programas, como `cd` que está embutido no shell.

Estes comandos vem em pacotes, a maioria da Free Software Foundation (ou GNU). Em vez de listar estes pacotes aqui, vou te encaminhar para o *Linux From Scratch HOWTO* `http://www.linuxfromscratch.org`. Este tem uma lista completa e atualizada dos pacotes que vão em um sistema Linux assim como as instruções de como os construir.

13 Conclusão

Uma das melhores coisas sobre o Linux, em minha humilde opinião, é que você pode entrar dentro dele e ver como ele realmente funciona. Espero que tenha gostado tanto quanto eu. E espero também que estas poucas notas tenham te ajudado.

14 Administrivia

14.1 Copyright

Licence http://www.gnu.org/copyleft/gpl.html . Por favor, me cite como autor se usar todo ou partes deste documento em outro documento.

14.2 Página Web

A versão mais recente deste documento está em *From Powerup To Bash Prompt* http://www.netspace.net.au/~gok/power2bash assim como seu companheiro "Building a Minimal Linux System from Source Code".

Há uma tradução para o Francês em *From Powerup To Bash Prompt* http://www.freenix.fr/unix/linux/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html obrigado a Dominique van den Broeck. Logo haverá uma para o Japonês feita por Yuji Senda, se não já estiver pronta em *Japanese Documentation and FAQ Project* http://www.linux.or.jp/JF

14.3 Retorno

Gostaria de ouvir quaisquer comentários, críticas ou sugestões de melhoras que você tenha. Por favor, envie-as para *Greg O'Keefe* mailto:gcokeefe@postoffice.utas.edu.au

14.4 Agradecimentos

Os nome de produtos são marcas registradas de seus respectivos proprietários, e por meio disto considerado adequadamente reconhecidos.

Existem algumas pessoas que quero agradecer, por ajudar que isto acontecesse.

Michael Emery

Por me lembrar sobre o Unios

Tim Little

Por várias boas dicas sobre `/etc/passwd`

sPaKr on #linux in efnet

Quem me avisou que o syslogd precisa de `/etc/services`, e me apresentou a frase "rolling your own" para descrever a construção de um sistema a partir do código fonte.

Alex Aitkin

Por me chamar a atenção sobre Vico e seu "verum ipsum factum" (o conhecimento vem pela prática).

Dennis Scott

Por corrigir minha aritimética hexadecimal.

jdd

Por me mostrar alguns erros tipográficos.

David Leadbeater

Por contribuir com vários assuntos "tortuosos" sobre os daemons do kernel.

Dominique van den Broeck

Por traduzir este documento para o Francês.

Matthieu Peeters

Por várias boas informações sobre os daemons do kernel.

John Fremlin

Por várias boas informações sobre os daemons do kernel.

Yuji Senda

Pela tradução para o Japonês.

Antonius de Rozari

Por contribuir com uma versão do UNIOS para o montador GNU as (veja a seção de recursos na página web)

Botp Peña

Pelo link para o "roll your own os".

Kees J. Bot

Autor das páginas de manual do Minix. Em particular, a página de manual sobre o `boot` que tem o subtítulo "boot - from power on to the login prompt" (de quando é ligado até o login). Apenas descobri esta pequena pérola depois que escrevi e submeti o documento atual para o LDP.

Scott Hankin

For spotting a typo. ??

14.5 Histórico de modificações

14.5.1 0.9 -> 0.9a (Novembro 2000)

- Adicionado o link "roll your own os" à seção Hardware.

14.5.2 0.8 -> 0.9 (Novembro 2000)

- Incorporada várias informações de Matthieu Peeters e John Fremlin em daemons do kernel e o sistema de arquivos `/proc`.

14.5.3 0.7 -> 0.8 (Setembro 2000)

- Removidas as instruções sobre como construir um sistema, colocando-as em um documento separado. Ajustado alguns links adequadamente.
- Movida a página web de *learning@TasLUG* `http://learning.taslug.org.au/power2bash` para *meu próprio espaço na web* `http://www.netspace.net.au/~gok/power2bash` .
- Falhei completamente em incorporar um monte de bom material contribuído por várias pessoas. Talvez da próxima vez :(

14.5.4 0.6 -> 0.7

- Mais ênfase nas explicações, menos em como construir um sistema, informações de construção reunidas em uma seção separada e reduzida, encaminhar os leitores para o "Linux From Scratch" de Gerard Beekmans para construções sérias.
- Adicionada contribuições de David Leadbeater.
- Corrigido um par de urls, adicionado link para baixar o unios em learning.taslug.org.au/resources.
- Testadas e corrigidas urls.
- Reescrita geral, posto em ordem.

14.5.5 0.5 -> 0.6

- Adicionado o histórico de modificações.
- Adicionada várias tarefas a fazer.

14.6 TODO (Tarefas a fazer)

- Explicar módulos do kernel, depmod, modprobe, insmod e tudo mais (vou precisar aprender primeiro!).
- Mencionar o sistema de arquivos /proc, potencial para exercícios.
- Converter para docbook sgml
- Adicionar mais exercícios, talvez uma seção inteira de grandes exercícios, como criar um sistema de arquivos mínimo com arquivos da instalação de uma distribuição.
- add makefile hack to bash build instructions - see easter notes. ??