CENTRO UNIVERSITÁRIO DE BRASÍLIA

FACULDADE DE CIENCIA DA COMPUTAÇÃO

CENTRO UNIVERSITÁRIO DE BRASÍLIA

FACULDADE DE CIENCIA DA COMPUTAÇÃO

Gabriel de Aragão Esteves

Trabalho de Escrever em detalhes todo o passo a passo do processo de construção da imagem do sistema operacional

Sumário

Requisitos

Os requisitos básicos usados para produzir uma remasterização linux a partir de um Debian são:

Superuser (root)

Uma versão atualizada do live-build

A POSIX-compliant shell, tanto bash quanto dash

debootstrap

Linux 2.6 or newer.

COMO FOI INICIADO

Foi Criado uma imagem padrão

Para usar esses produtos, você precisa de um sistema para construí-los que atenda aos requisitos listados em Requisitos e tenha o live-build instalado conforme descrito em Instalando o live-build.

Examine o conteúdo do diretório config / se desejar. Você verá armazenada aqui uma configuração esquelética, pronta para customizar ou, neste caso, usar imediatamente para construir uma imagem padrão.

Agora, como superusuário, construa a imagem, salvando um registro enquanto constrói com tee.

Uma utilidade Web

Nossa escolha de LXDE para este exemplo reflete nosso desejo de fornecer um ambiente de desktop mínimo, já que o foco da imagem é o único uso que temos em mente, o navegador da web. Poderíamos ir ainda mais longe e fornecer uma configuração padrão para o navegador da web em config / includes.chroot / etc / iceweasel / profile / ou pacotes de suporte adicionais para visualizar vários tipos de conteúdo da web, mas deixamos isso como um exercício para o leitor .

Uma imagem Personalizada

Crie um projeto para construir uma imagem personalizada, contendo seu software favorito para levar com você em um pendrive aonde quer que vá, e evoluindo em revisões sucessivas conforme suas necessidades e preferências mudam.

Já que iremos mudar nossa imagem personalizada ao longo de uma série de revisões, e queremos rastrear essas mudanças, tentando coisas experimentalmente e possivelmente revertendo-as se as coisas não funcionarem, vamos manter nossa configuração no popular sistema de controle de versão git. Também usaremos as melhores práticas de configuração automática por meio de scripts automáticos, conforme descrito em Gerenciando uma configuração.

A PRIMEIRA REVISÃO

```
$ mkdir -p tutorial3/auto
$ cp /usr/share/doc/live-build/examples/auto/* tutorial3/auto/
$ cd tutorial3
```

Foi editado `auto/config` para ler desse modo:

```
#!/bin/sh

lb config noauto \
    --architectures i386 \
    --linux-flavours 686-pae \
    "${@}"
```

Foi executado lb config para gerar a árvore de configuração, usando o script auto / config que foi criado

Então foi preenchido a lista de pacotes locais:

```
$ echo "task-lxde-desktop iceweasel xchat" >> config/package-lists/my.list.chroot
```

```
$ echo "task-lxde-desktop iceweasel xchat" >> config / package-lists / my.list.chroot
```

Primeiro foi feito, --architectures i386 garante que em nosso sistema de compilação amd64, construamos uma versão de 32 bits adequada para uso na maioria das máquinas.

Em segundo lugar, usamos --linux-flavors 686-pae porque não prevemos o uso desta imagem em sistemas muito mais antigos.

Terceiro, escolhemos o metapacote de tarefas lxde para nos dar uma área de trabalho mínima. E, finalmente, adicionamos dois pacotes favoritos iniciais: iceweasel e xchat

Então foi construída a imagem:

```
# lb build
```

2 REVISÃO

Nesta revisão, foi feito uma limpeza desde a primeira construção, adicionado o pacote vlc à nossa configuração, reconstruido, testado e confirmado.

O comando lb clean limpará todos os arquivos gerados da compilação anterior, exceto o cache, o que evita a necessidade de baixar novamente os pacotes. Isso garante que a construção lb subsequente executará novamente todos os estágios para regenerar os arquivos de nossa nova configuração.

```
# lb clean
```

Então foi anexado o pacote vlc à nossa lista de pacotes locais em config/package-lists/my.list.chroot:

```
$ echo vlc >> config/package-lists/my.list.chroot
```

Foi testado e, quando estivamos satisfeitos, enviamos a próxima revisão:

```
$ git commit -a -m "Adding vlc media player."
```

Claro, mudanças mais complicadas na configuração são possíveis, talvez adicionando arquivos em subdiretórios de config/.Quando você confirma novas revisões, apenas tome cuidado para não editar manualmente ou confirmar os arquivos de nível superior em config. Contando LB_* variáveis, visto que também são produtos de construção e são sempre limpos por lb clean. e recriado com lb config

VNC KIOSK CLIENT

Criamos uma imagem com o live-build para inicializar o servidor diretamente em um servidor VNC

Foi feito um diretório de construção e criado uma configuração esquelética dentro dele, desabilitando recomenda fazer um sistema mínimo. E então crie duas listas de pacotes iniciais: a primeira gerada com um script fornecido por live-build chamado Packages e a segunda incluindo xorg, gdm3, metacity e xvnc4viewer.

```
$ mkdir vnc-kiosk-client
$ cd vnc-kiosk-client
$ lb config -a i386 -k 686-pae --apt-recommends false
$ echo '! Packages Priority standard' > config/package-lists/standard.list.chroot
$ echo "xorg gdm3 metacity xvnc4viewer" > config/package-lists/my.list.chroot
```

Depois disso, criamos o diretório / etc / skel em config / includes.chroot e colocamos uma .xsession personalizada nele para o usuário padrão que iniciará o metacity e iniciamos o xvncviewer, conectando-se à porta 5901 em um servidor em 192.168.1.2:

```
$ mkdir -p config/includes.chroot/etc/skel
$ cat > config/includes.chroot/etc/skel/.xsession << EOF
#!/bin/sh

/usr/bin/metacity &
/usr/bin/xvncviewer 192.168.1.2:1

exit
EOF
```

Uma imagem de base para uma chave USB de 128 MB

No Ceub Os, cortamos apenas o suficiente para abrir espaço para material adicional dentro de um tamanho de mídia de 128 MB, mas sem fazer nada para destruir a integridade dos pacotes contidos nele, como a eliminação de dados de localidade através do pacote localepurge ou outro essas otimizações "intrusivas". Em particular, usamos `--debootstrap-options` para criar um sistema mínimo do zero

```
$ lb config --apt-indices false --apt-recommends false --debootstrap-options "--variant=minbase" --firmware-chroot false --memtest none
```

fizemos isso para a imagem funcionar corretamente, adicionamos novamente, pelo menos, dois pacotes recomendados que foram deixados de fora pela opção --apt-recommends false. Consultando e Ajustando o APT para economizar espaçoE, finalmente, --memtest none impede a instalação de um testador de memória.

Uma área de trabalho GNOME localizada e um instalador

Nosso problema inicial é a descoberta dos nomes das tarefas de linguagem apropriadas. Atualmente, o live-build não pode ajudar com isso. Embora possamos ter sorte e descobrir isso por tentativa e erro, existe uma ferramenta, grep-dctrl, que pode ser usada para extraí-la das descrições de tarefas em tasksel-data, para se preparar, certifique-se de ter ambos dessas coisas:

```
# apt-get install dctrl-tools tasksel-data
```

Agora podemos pesquisar as tarefas adequadas, primeiro com:

```
$ grep-dctrl -FTest-lang de /usr/share/tasksel/descs/debian-
tasks.desc -sTask
Task: german
```

Por meio desse comando, descobrimos que a tarefa é chamada, de maneira bastante clara, de alemã. Agora, para encontrar as tarefas relacionadas:

```
$ grep-dctrl -FEnhances german
/usr/share/tasksel/descs/debian-tasks.desc -sTask
Task: german-desktop
Task: german-kde-desktop
```

No momento da inicialização, geraremos o local de_CH.UTF-8 e selecionaremos o layout de teclado ch. Agora vamos juntar as peças. Lembrando de Usar metapacotes que os metapacotes de tarefa são prefixados como tarefa-, apenas especificamos esses parâmetros de inicialização de idioma e, em seguida, adicionamos os pacotes de prioridade padrão e todos os nossos metapacotes de tarefas descobertos à nossa lista de pacotes da seguinte maneira:

```
$ mkdir live-gnome-ch
$ cd live-gnome-ch
$ lb config \
     -a i386 \
     --bootappend-live "boot=live components
locales=de_CH.UTF-8 keyboard-layouts=ch" \
     --debian-installer live
$ echo '! Packages Priority standard' > config/package-
lists/standard.list.chroot
$ echo task-gnome-desktop task-german task-german-desktop >>
config/package-lists/desktop.list.chroot
```

```
$ echo debian-installer-launcher >> config/package-
lists/installer.list.chroot
```

Observe que incluímos o pacote debian-installer-launcher para iniciar o instalador a partir da área de trabalho ativa. O tipo de kernel 586, que atualmente é necessário para que o inicializador funcione corretamente, será incluído por padrão.