IBM

developerWorks 中国 技术主题 Linux 文档库

理解 chroot

通过编写 chroot 来认识 chroot 发挥的作用和它带来的好处

chroot 在 Linux 系统中发挥了根目录的切换工作，同时带来了系统的安全性等好处。本文通过编写 chroot 来理解 chroot 的作用和好处，这不仅有助于更好的使用 chroot，同时加深了对 Linix 系统初始 RAM 磁盘工作的认识。

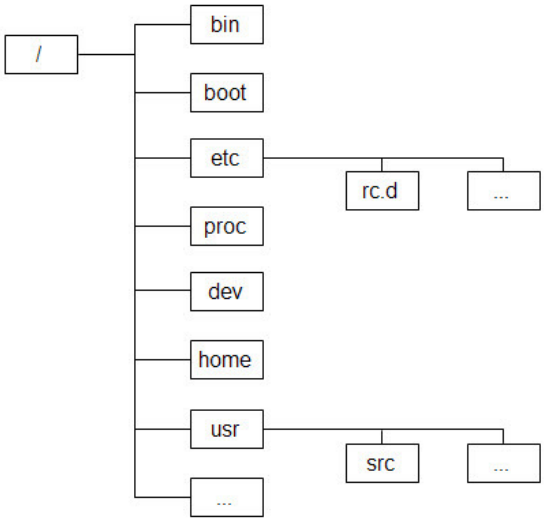
王华东，自由职业者，熟悉 Linux 系统管理，对 Linux 和 Open source 有浓厚的兴趣。通过 wstoneh@126.com 可以和他联系。

2009 年 7 月 09 日

什么是 chroot

chroot，即 change root directory (更改 root 目录)。在 linux 系统中，系统默认的目录结构都是以 `/` ，即是以根 (root) 开始的。而在使用 chroot 之后，系统的目录结构将以指定的位置作为 `/` 位置。

图 1. Linux 系统的目录结构



为何使用 chroot

在经过 chroot 之后，系统读取到的目录和文件将不在是旧系统根下的而是新根下(即被指定的新的位置)的目录结构和文件，因此它带来的好处大致有以下3个：

1. 增加了系统的安全性，限制了用户的权力；
   在经过 chroot 之后，在新根下将访问不到旧系统的根目录结构和文件，这样就增强了系统的安全性。这个一般是在登录 (login) 前使用 chroot，以此达到用户不能访问一些特定的文件。
2. 建立一个与原系统隔离的系统目录结构，方便用户的开发；
   使用 chroot 后，系统读取的是新根下的目录和文件，这是一个与原系统根下文件不相关的目录结构。在这个新的环境中，可以用来测试软件的静态编译以及一些与系统不相关的独立开发。
3. 切换系统的根目录位置，引导 Linux 系统启动以及急救系统等。

chroot 的作用就是切换系统的根位置，而这个作用最为明显的是在系统初始引导磁盘的处理过程中使用，从初始 RAM 磁盘 (initrd) 切换系统的根位置并执行真正的 init。另外，当系统出现一些问题时，我们也可以使用 chroot 来切换到一个临时的系统。

chroot 的使用

为了更好的理解 chroot 发挥的作用，我们将尝试指定一个特定的位置进行根目录切换。但是由于在经过 chroot 之后，系统读取到的 bin/ 等与系统相关目录将不再是旧系统根目录下的，而是切换后新根下的目录结构和文件，因此我们有必要准备一些目录结构以及必要的文件。

清单 1. 准备切换的目录结构

```
$ pwd
/home/wstone/Build/work
$ tree .
.
|-- bin
| |-- ash -> busybox
| |-- bash
| `-- busybox
|-- etc
`-- newhome
```

busybox
Busybox 被称为是嵌入式 Linux 中的瑞士军刀。Busybox 包含了许多有用的命令，如 cat、find 等，但是它的体积却非常的小。

这里使用了静态编译后的 busybox 来提供必要的命令，使用静态编译仅是为了避免动态库文件的拷贝。当然我们也可以拷贝旧系统的下的命令到新的目录结构中使用，但是那些命令通常是动态编译的，这就意味着我们不得不拷贝相关的动态库文件到相应的目录结构中。同时这里的 bash 也非真正的 Bourne Again shell，而是一个执行 ash 的 shell 脚本。在清单 2中，展示了位于旧系统中的 chroot 命令的使用。需要注意的是在使用 chroot 时，要求拥有相关的操作权限。

清单 2. 位于系统中的 chroot 的使用

```
$ pwd
/home/wstone/Build/work

# chroot .
# pwd
/

# ls
ash: ls: not found

# busybox ls
bin      etc      newhome

3 directories, 3 files
```

我们可以看到当前路径(/home/wstone/Build/work/)，在经过 chroot 后转变成了 `/` 目录，同时从新根下读取了与系统相关的目录结构。使用 `ls` 命令失败是由于我们创建的测试目录结构中并没有包含命令 `ls`，但是我们成功的使用了 busybox 中的 `ls`。以上看到的只是 chroot 的一种使用方式，其实标准的 chroot (Coreutils - GNU core utilities 提供的 chroot)使用方式有2种：

清单 3. 标准 chroot 的2种使用方式

```
[1] chroot NEWROOT [COMMAND...]
[2] chroot OPTION
```

刚才我们使用的是方式[2]。这将在没有给定环境时，默认执行 `/bin/sh`，但是当给定环境后，将运行 `${SHELL} -i`，即与环境相同的可交互的 shell。我们的目录结构中并没有包含sh，显然清单 2中的 chroot 运行了 `${SHELL} -i`。当然我们也可以在进行切换时指定需要的命令，即使用方式[1]。

清单 4. chroot 另一种方式的使用

```
# chroot . /bin/ash
#
```

在清单 4 中，尝试了在经过 chroot 后，执行新目录结构下的 ash shell。不得不说的是，如果新根下的目录结构和文件准备的够充分，那么一个新的简单的 Linux 系统就可以使用了。其实更为常见的是在初始 RAM 磁盘 (initrd)中使用 chroot，以此来执行系统的 `init`。清单 5 中，展示的是在 Linux 2.4 内核 initrd 中使用 chroot。

清单 5. 在 Linux 2.4 内核 initrd 中使用 chroot 的示例

```
mount /dev/hda1 /new-root
cd /new-root
pivot_root . old-root
exec chroot . /sbin/init <dev/console >dev/console 2>&1
umount /old-root
```

由于 Linux 内核的升级，initrd 处理机制和格式发生了变化，在 Linux 2.6 内核 initrd 中不能再使用 pivot_root，因此一般也不再使用 chroot，而是选择使用 busybox 提供的 switch_root 或者 klibc 提供的 run-init 进行根目录的切换。(这并不是说不能在 Linux 2.6内核 initrd 中使用 chroot，选择 switch_root 或 run-init 仅是出于习惯和方便的考虑。)但是实质上，它们仅是将 chroot 的功能进行了封装，以此更加方便简单的切换根目录。

清单 6. 在 Linux 2.6 内核 initrd 中 chroot 的使用

```
[1] find -xdev / -exec rm '{}' ';
[2] cd /newmount; mount --move . /; chroot .
```

switch_root 和 run-init 完成了类似清单 6中的功能，删除 rootfs 的全部内容以释放空间，以及挂载新的根文件系统并进行切换。在 busybox 和 klibc中也有提供 chroot 命令，只是功能上与 Coreutils (GNU core utilities) 包含的 chroot 有稍许差异。

编写一个 chroot

上面介绍了 chroot 及其使用，但是编写一个简单的 chroot 并不复杂，下面我们就尝试编写chroot 以此来更好的认识 chroot 的处理过程，先编写一个粗略的 chroot 然后再完善它的功能。chroot 的编写涉及了2个函数，chroot() 以及 chdir()，它们都包含在 unistd.h 头文件中。

清单 7. 编写 chroot 涉及的2个函数

```
#include <unistd.h>
int chroot(const char *path);
int chdir(const char *path);
```

chroot() 将切换参数 path 所指位置为根目录 (/)，chdir() 用来将当前的工作目录改变成以参数path 所指的目录。以此我们可以编写一个非常粗略的 `chroot`。

清单 8. 粗略的 `chroot`

```
#include <unistd.h>

int main(int argc, char *argv[])
{
    chroot(".");
    chdir("/");

    char *arrays[]={"ash",NULL};
    execvp("ash", arrays);

    return 0;
}
```

这个粗略的 `chroot` 仅能切换当前位置为根目录，同时默认执行 ash shell，不包含任何的错误处理及警告。编写并保存代码为 `test.c`。在清单 9 中，展示了这个*粗略 `chroot`* 的使用情况，成功的进行了根目录的切换。

清单 9. 粗略 `chroot` 的使用

```
$ gcc -Wall test.c -o test

# ./test
# ls
ash: ls: not found

# busybox ls
bin      etc      newhome  test     test.c
```

下面给出功能将近完整的 chroot ，加上了一些错误处理并新增了可执行指定命令的功能。当在没有给出 chroot 切换后要执行的命令时，默认执行 `/bin/sh`，同时检测环境以确认使用何种 shell。

清单 10. 功能完整的 chroot

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    if(argc<2){
        printf("Usage: chroot NEWROOT [COMMAND...] \n");
        return 1;
    }

    printf("newroot = %s\n", argv[1]);
    if(chroot(argv[1])) {
        perror("chroot");
        return 1;
    }

    if(chdir("/")) {
        perror("chdir");
        return 1;
    }

    if(argc == 2) {
        argv[0] = getenv("SHELL");
        if(!argv[0])
            argv[0] = (char *)"/bin/sh";

        argv[1] = (char *) "-i";
        argv[2] = NULL;
    } else {
        argv += 2;
    }

    execvp (argv[0], argv);
    printf("chroot: cannot run command `%s`\n", *argv);

    return 0;
}
```

保存以上代码为 newchroot.c 文件，编译后运行测试其功能。最后要指出的是，本文中的 `chroot` 并没有使用静态编译。如果有必要(如，在 initrd 中使用 chroot)，chroot 应该使用静态编译，若是使用动态编译，那么要拷贝相关的动态库文件到相应目录结构中。

清单 11. `newchroot` 的测试

```
$ gcc -Wall newchroot.c -o newchroot

# ./newchroot . /bin/ash
newroot = .
#
```

结束语

在 Linux 系统初始引导的过程中，通常都有使用 chroot。但是 chroot 的好处不仅于此，它还增加了系统的安全性等。而通过本文后半部分对 chroot 的认识，我相信读者可以更好的发挥chroot 的作用。

下载

描述	名字	大小
样例代码	work.tar.bz2	734KB

参考资料

IBM Bluemix 资源中心

参考 wiki 上关于 chroot 的介绍。

查阅与 chroot 相关的 man 手册。

查看文章“BusyBox 简化嵌入式 Linux 系统”，了解更多有关 busybox 的内容。

查看文章“Linux initial RAM disk (initrd) overview”，获得更多有关 initrd 的知识。

查看文章“Linux2.6 内核的 Initrd 机制解析”，了解更多有关 Linux 2.4 内核和 2.6 内核的 initrd 的内容。

查看 Linux 内核中“Using the initial RAM disk (initrd)”和“ramfs, rootfs and initramfs”文档。

可以从 Coreutils - GNU core utilities 中找寻到标准的 chroot 源码。

Busybox 和 klibc 中也有包含有关 chroot 的源码。

在 developerWorks Linux 专区 寻找为 Linux 开发人员（包括 Linux 新手入门）准备的更多参考资料，查阅我们 最受欢迎的文章和教程。

在 developerWorks 上查阅所有 Linux 技巧 和 Linux 教程。