fs-utils: File Systems Access Tools in Userland

Arnaud Ysmal and Antti Kantee

EuroBSDCon 2009
Cambridge, UK
September 2009

Outline

Outline

- **Presentation**
- Development
- Perspectives

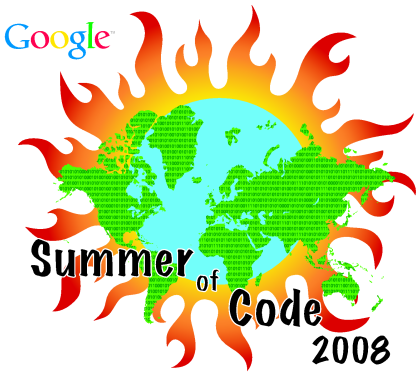
What is fs-utils ?

- File system access utilities

fs-utils is a mtools-like set of tools:

- Set of tools to access file systems
- Runs completely in userspace
- Uses unmodified file system code from NetBSD
- Not limited to NetBSD

Origin



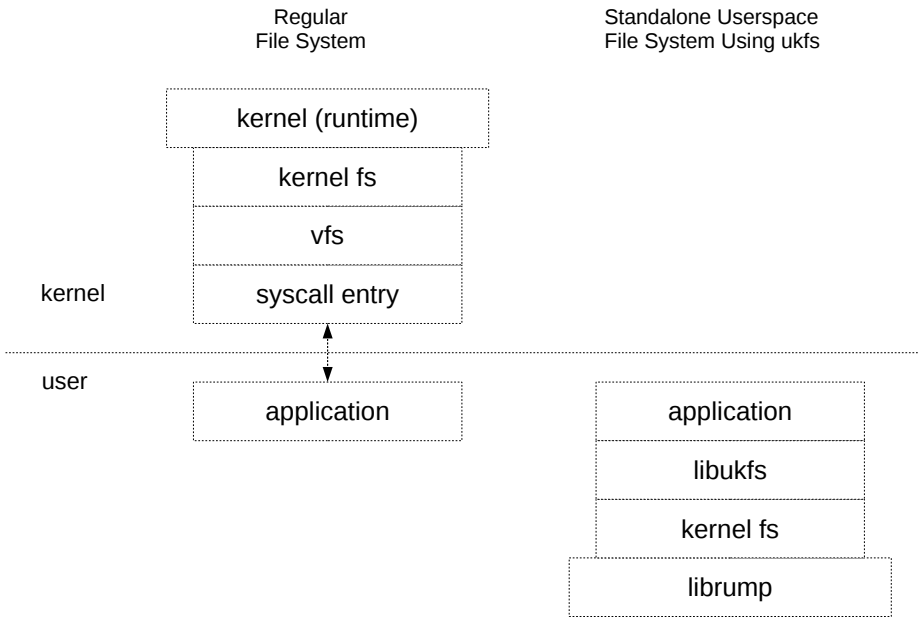
- Google Summer Of Code 2008 & The NetBSD Foundation
- File system access utilities (generalized mtools)

Use cases

- Accessing file system images
  `fsu_rm an_image -R /stand/i386/5.99.15`
- Accessing block device
  `fsu_ls /dev/wd1a /root`
- Testing file system compiled as rump library
  - Testing new file system code
  - Testing new VFS code

RUMP: Runnable Userspace Meta Program

- Runs kernel code in userland



Two ways of using the file system code from the NetBSD kernel

UKFS: User-Kernel File System

- Handles mount information
- Allows file system images access

```
fs = ukfs_mount("ffs", "/ffs.img", "/", 0,
    &args, sizeof(args));
ukfs_mkdir(fs, "/dir", 0777);
ukfs_read(fs, "/a/file", 0, buf, 11);
ukfs_release(fs, 0);
```

Outline

- Presentation
- **Development**
- Perspectives

Libraries

fs-utils contains two libraries:

- fsu_mount
- fsu_utils

fsu_mount

- Generic file system mounting library
- Automatic file system type detection

- Eases “virtual” mounting
- Handles file system specific code

fsu_utils

- Handles files access (fopen, fread, ...)
- Handles directories access (opendir, ...)
- Handles hierarchy traversing (fts)

Types of tools

fs-utils contains three kind of tools:

- Patched commands (chown, chmod, ls, ...)
- Rewritten commands (cat, cp, ...)
- New commands (console, ecp, write)

Patched commands

- Declaring a struct ukfs
  `DECLARE_UKFS(ukfs)`
- Mounting the image
  `FSU_MOUNT(argc, argv, ukfs);`
- Using ukfs or fsu_utils to access files/directories
  `#define stat(file, sb) ukfs_stat(ukfs, file, sb)`
- Adding a proper usage

`chflags, chmod, chown, cp, du, ln, ls,`
`mkdir, mkfifo, mknod, rm, rmdir`

Rewritten commands

- License limitation
  `diff`
- RUMP limitation (e.g. no fork)
  `cat, mv, ...`

New commands

- fsu_exec: executing local commands on rump file system
  `fsu_exec an_img ${EDITOR} /a_file`
- fsu_write: writing the output of a program to a file in the image
  `ls ~ | fsu_write an_img /a_file`
- fsu_ecp
- fsu_console

fsu_ecp

fsu_ecp is also aliased as fsu_put and fsu_get

- Allows copying files between rump fs and host fs

```
fsu_put an_image -R /rescue /
fsu_ecp an_image -gR /root/result ~/
```

- fsu_put can be used to implement makefs

fsu_console

fsu_console provides shell-like access to file systems

```
$ fsu_console /my.img
/my.img(ffs):/ # ls
boot i386 miniroot.kmod netbsd
/my.img(ffs):/ # put /boot.cfg .
/my.img(ffs):/ # ls
boot boot.cfg i386 miniroot.kmod netbsd
/my.img(ffs):/ # cd /etc
/my.img(ffs):/etc # exec vi rc.conf
/my.img(ffs):/etc # exec grep hostname rc.conf
hostname="foobar"
```

Supported file systems

- block device based file systems
  `cd9660, efs, ext2, hfs, ffs, fat, lfs, ntfs, sysvbfs, udf`
- memory based file systems
  `tmpfs`
- network based file systems
  `nfs`
- fuse/refuse based file systems
  `e.g. sshfs, ntfs-3g`

NetBSD

How to install fs-utils under NetBSD:

- Use the filesystems/fs-utils package from pkgsrc
- Build from source by doing a checkout of NetBSD's othersrc repository

Other operating systems

How to make it work ?

1. Make build work
2. Solve ABI mismatch issues

Outline

- Presentation
- Development
- **Perspectives**

Benefits

- Genericity
  - File system independent tools
  - Enables access to rump supported file systems
  - Does not need a specific kernel or specific options in the kernel
  - Uses nearly the same syntax as standard tools

Benefits

- Security
  - Runs in userspace
  - Uses robust code (kernel)
  - Does not need to be root
- Development
  - Eases file system development and debugging
  - Eases VFS development and debugging

Future work

- Adding commands to the NetBSD base system
- Merging rump access and system call access
- Adding support on other operating systems

Conclusion

- Provides access/modification on a file system
- Supports more than 12 file systems
- Does not require specific kernel options
- Does not need to be root
- Runs completely in userland
- Reuses kernel code via rump
- Reuses utility code

- http://NetBSD.org/docs/rump/
- http://NetBSD.org/docs/rump/ukfs.html
- http://NetBSD.org/~stacktic/fs-utils.html
- http://NetBSD.org/contrib/soc-projects.html
- http://code.google.com/soc