ファイルシステム

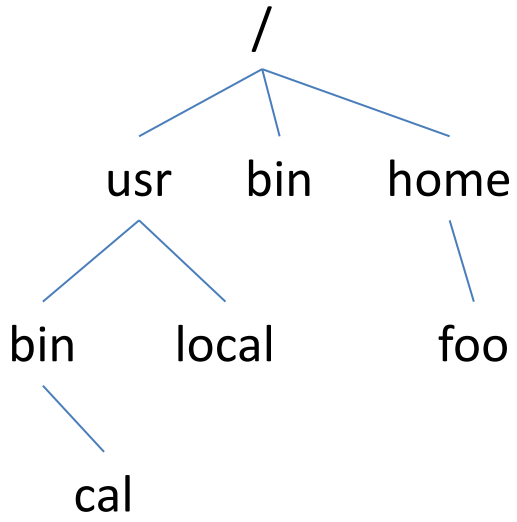
コンピュータリテラシ

2016年5月2日

建部修見

ファイルシステム

- ファイルを管理するシステム
  - ハードディスク、USBなど
- 階層的な名前空間
  - UNIXは/（ルートディレクトリ）から始まる単一な名前空間
  - ディレクトリ、ファイル
  - /usr/bin/cal
- アクセス制御
  - 所有者、所有グループ、その他
  - 読込可、書込可、実行可



リモートファイルシステム

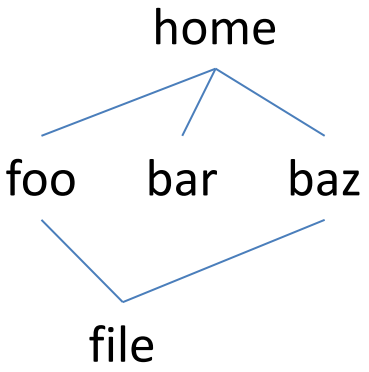
- ネットワークで接続したファイルサーバが管理するファイルシステム
  - Network File System (NFS; 1984 Sun Microsystems), CIFS (Microsoft)
- 複数の計算機でファイルシステムを共有
- マウント
  - 他のファイルシステムを使えるようにする操作
  - マウントポイント
  - /home

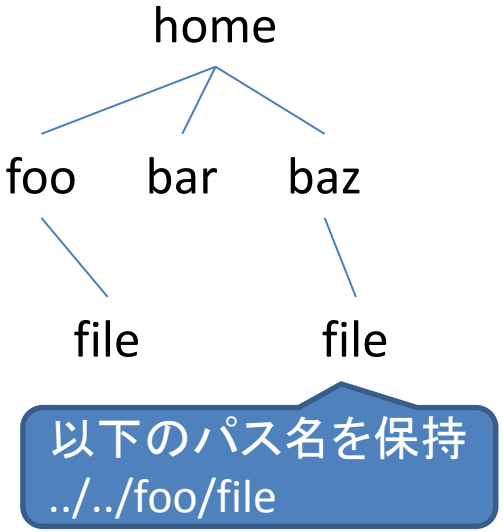
パス名

- ディレクトリ名を”/”で区切る
  - Windowsは”¥”
- 絶対パス（/で始まる）
  - /usr/bin/cal
- 現在のディレクトリからの相対パス
  - bin/cal
- 現在のディレクトリは”.”、親ディレクトリは”..”で表す
  - ../bin/cal, ../../bin/cal, ./bin/cal

リンク

- 同一のファイル（エントリ）を指す
- ハードリンク
  - 同一エントリへのリンク
  - リンクカウント
  - ファイルを移動してもリンクは切れない
  - 同一ファイルシステム内しかリンクできない
- シンボリックリンク
  - 別エントリ
  - リンク先エントリへのパス名を保持
  - ファイルを移動するとリンク切れ
  - 別ファイルシステムにもリンク可能





アクセス制御

- ファイルやディレクトリに対するアクセス制御
- UNIXでは基本的にはuser（所有者）、group（グループ）、other（それ以外）に対しアクセス制御
  - 読込可/不可、書込可/不可、実行可/不可
  - ディレクトリの実行権限はディレクトリへのアクセス可/不可
- setuid (set user ID on execution), setgid (set group ID on execution)
  - 所有者、グループの権限でプログラムを実行
- sticky bit（ディレクトリに対する特別なアクセス制御）
  - 所有者しか移動、削除不可
- アクセス制御リスト（ACL）でより詳細に制御
  - 特定ユーザ、特定グループに対しアクセス制御
  - Mac OS Xでは削除可/不可、アペンド可/不可なども

拡張属性

- 拡張属性名と値
- Linuxでは属性名に名前空間があり、user名前空間は設定可能
- Mac OS Xは名前空間はない

典型的なUNIXのディレクトリ構造

- /bin – 最低限の実行プログラム
- /dev – デバイスファイル
- /etc – 各種設定ファイル
- /home – 各自のホームディレクトリ
- /sbin – 最低限の管理用実行プログラム
- /tmp – 一時ファイル
- /usr – ユーザ共通のプログラムなど
- /var – ログファイルなど

COINSのマウント状況

```
$ df -PH
Filesystem                                    Size   Used  Avail Capacity  Mounted on
/dev/disk0s6                                  200G   124G    75G    63%    /
devfs                                         190k   190k     0B   100%    /dev
map -hosts                                      0B     0B     0B   100%    /net
map auto_nfs                                    0B     0B     0B   100%    /home
map auto_nfs                                    0B     0B     0B   100%    /usr/local3
pentas-fs.coins.tsukuba.ac.jp:/vol0/home      3.7G   692M   3.0G    19%    /home
pentas-fs.coins.tsukuba.ac.jp:/vol0/local3    107G    23G    85G    22%    /usr/local3
```

- /はローカルディスク（/dev/disk0s6）
- /homeと/usr/local3はpentas-fsをマウント
  - 全計算機がマウントしているためどの計算機でも同一ファイルにアクセス可能

ホームディレクトリ

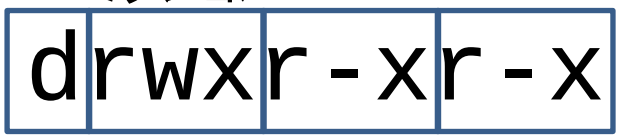
- loginしたときのディレクトリ
- ユーザが読み書き可能なディレクトリ
- ~（チルダ）で表される

ディレクトリリスティングとパミッション

- lsコマンド – ディレクトリリスティング

```
$ ls -l
total 23626
drwx------   2 tatebe  prof      6144 Apr 18 17:16 Desktop
drwx------   2 tatebe  prof      2048 Apr 18 17:25 Documents
drwx------   2 tatebe  prof      1024 Apr 26 16:42 Downloads
drwx------@ 25 tatebe  prof      1024 Apr 19 20:26 Library
drwx------   8 tatebe  prof      1024 Apr 26 18:00 Mail
drwx------  10 tatebe  prof      1024 Apr 26 17:33 Maildir
drwxr-xr-x   2 tatebe  prof        80 Apr 18 17:22 Music
...
```

パミッション



r – 読込可
w – 書込可
x – 実行可

d – ディレクトリ
- – ファイル

-aオプション

```
$ ls -la
total 23760
drwxr-xr-x  38 tatebe  prof       4096 Apr 30 11:23 .
drwxr-xr-x  98 root    wheel      4096 Apr 12 10:03 ..
-rw-r--r--   1 tatebe  prof          5 Feb 25  2014 .CFUserTextEncoding
-rw-------   1 tatebe  prof        930 Apr 18 17:22 .ICEauthority
drwx------   2 tatebe  prof       2048 Apr 13 11:43 .Trash
-rw-------   1 tatebe  prof        130 Apr 13 12:29 .Xauthority
-rw-------   1 tatebe  prof       4096 Apr 13 11:30 ._Library
drwxr-xr-x   2 tatebe  prof         80 Apr 13 11:52 .abrt
-rw-------   1 tatebe  prof       6502 Apr 30 11:25 .bash_history
-rw-r--r--   1 tatebe  prof        164 Feb 25  2014 .bash_logout
-rw-r--r--   1 tatebe  prof        218 Feb 25  2014 .bash_profile
-rw-r--r--   1 tatebe  prof        161 Apr 19 17:33 .bashrc
...
```

パミッション　所有者グループ サイズ　修正日時　　エントリ名

リンクカウント

パミッション変更

- chmodコマンド – パミッション変更

$ chmod g+w entry　　#groupに書込許可

$ chmod o+w entry　　#otherに書込許可

$ chmod u-w entry　　#userに書込不許可

$ chmod 755 entry　　#rwxr-xr-xに変更

パミッションを8進数で表現　7　5　5

111 101 101

ファイル、ディレクトリ操作

コマンド	操作	コマンド例
cp src dest	ファイルコピー	cp a b
cp src … dest-dir	複数ファイルをコピー	cp a1 a2 a3 dir
mv src dest	ファイルを移動	mv a b
mv src … dest-dir	複数ファイルを移動	mv a1 a2 a3 dir
rm file …	ファイルを削除	rm a
mkdir dir …	ディレクトリ作成	mkdir foo
rmdir dir …	空ディレクトリ削除	rmdir foo
ln target linkname	ハードリンク作成	ln a b
ln -s target symlinkname	シンボリックリンク作成	ln -s a b

コピーとリンクの違い

コピーは同一ファイルがコピーされ、ストレージ容量を余分に必要とするが、
リンクは同一ファイルが参照され、余分なストレージ容量は必要ない

ファイル操作

コマンド	操作	コマンド例
cat file1 ...	ファイルを結合して出力	cat a
head file	ファイルの先頭10行（デフォルト）を出力	head a
tail file	ファイルを後ろ10行（デフォルト）を出力	tail a
echo string	指定された文字列を表示	echo hello, world
echo string > file	指定された文字列をファイルに書き込み	echo hello, world > file
echo string >> file	指定された文字列をファイルに追記（アペンド）	echo hello, world >> file

演習（１）

- 手引きの2.2節から2.7節を読もう
- ホームディレクトリに対しls、ls -l、ls -a、ls -alを実行してみよう
- オプションの意味をマニュアルページで確認しよう
- ディレクトリfooを作成してみよう
- cdコマンドでディレクトリを移動することができる。cd fooで先ほど作ったディレクトリに移動してみよう
- そこで、ls、ls -l、ls -a、ls -alを実行してみよう
- cdコマンドをディレクトリを指定しないで実行するとホームディレクトリに移動する。ホームディレクトリに移動してみよう

演習（2）

- whichコマンドで、実行するプログラムがどこに格納されているか分かる。which lsを実行して、lsコマンドの格納先を調べよう
- lsコマンドを先ほど作成したディレクトリfooにmylsという名前でコピーしてみよう
- 今コピーしたmylsコマンドを実行してみよう。実行するためには、パス名でmylsを指定する
  - ホームディレクトリにいる場合は、mylsはディレクトリfooにあるので、foo/mylsで指定できる
  - ディレクトリfooに移動した場合は、いまいるディレクトリは.で表されるので、./mylsで指定できる
  - いまいるディレクトリはpwdコマンドで分かる

演習（3）

- lsコマンドへのハードリンクls1をディレクトリfooに作成してみよう。エラーが出る場合はどのようなエラーが出るか？どうしてそのエラーが出るか考えてみよう
- mylsコマンドへのハードリンクls1をディレクトリfooに作成してみよう
- ls1を実行してみよう
- lsコマンドへのシンボリックリンクls2をディレクトリfooに作成してみよう
- ls2を実行してみよう

演習（４）

- mylsの実行権限を落としてみよう
- 実行権限がなくなっているか確認しよう
- mylsを実行するとどのようなエラーがでるか
- ディレクトリfooでls -laを実行して、リンクカウントを調べよう。ハードリンクしたファイルのリンクカウントが増えていることを確認しよう
- ls -laの出力を>（リダイレクト）を用いてファイルfoo.txtに書き出し、ファイルfoo.txtを提出しよう

オプション演習（１）

- ホームディレクトリでls -lを実行すると@が表示されるエントリがある。この意味を調べよう
- その内容を表示しよう
- ファイルを作成して、任意の拡張属性をつけてみよう

オプション演習（2）

- ディレクトリのリンクカウントは何を表しているか調べよう
  - そのために、ディレクトリを作成しリンクカウントを調べる
  - そのディレクトリにファイルを作成してリンクカウントを調べる
  - そのディレクトリにディレクトリを作成してリンクカウントを調べる

オプション演習（３）

- lsコマンドに-iオプションをつけるとiノード番号が表示される。iノードとはファイルやディレクトリの情報を格納するデータ構造であり、iノード番号はその番号である。iノード番号が同じであれば同一エントリ（同一ファイル）である
- mylsとls1のiノード番号が等しいことを確かめよう

オプション演習（４）

- 以下の内容のファイルgeteuid.cを作成しよう

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

main()
{
        printf("%d¥n", geteuid());
}
```

- このCプログラムをコンパイルして実行ファイルgeteuidを作成しよう

```
$ cc -o geteuid geteuid.c
```

- 実行してみよう
- 他の人に実行してもらおう
- setuid (set user ID on execution) bitをつけよう
- ls -lで確かめよう
- 他の人に実行してもらおう