目次

注： 今回の内容は，HDD(Hard Disk drive) 等の磁気ディスク装置の構造や記録方式等 (☆ 1) と大いに関連しているが，この授業ではこれらの内容については省略する．数理情報学科の学生は「情報処理の基礎／情報処理システム I」で学んでいることである（中野先生のウェブサイトで講義資料等を閲覧できる）．

☆ 1) シリンダ，トラック，セクタ，etc...

★1 ファイルシステム

OS は，補助記憶装置 (☆ 2) を扱いやすくするために，ファイル/ディレクトリという論理的な構造を与えて管理する (☆ 3)．このようなことを実現する仕組みを，**ファイルシステム**という (☆ 4)．補助記憶装置の使用開始時には一般に，その上にファイルシステムを構築する初期化作業（**フォーマット**）が必要となる．ファイルシステムが異なればファイルやディレクトリの形式等も異なるので，あるファイルシステム用にフォーマットされた補助記憶装置をコンピュータに接続しても，そこで稼働する OS がそのファイルシステムに対応していなければ読み書きできない．

☆ 2) 補助記憶装置: 二次記憶装置とも．PC では HDD であることが多い．

☆ 3) ［発展］ UNIX 系 OS では，補助記憶装置以外の入出力デバイスもファイルとして抽象化されている．

☆ 4) ［発展］ OS ではなくアプリケーション自身が専用のファイルシステムを構築することもある．

ファイルシステム: file system. フォーマット: format.

★1.1 ファイル

ファイルシステム上では，ひとまとまりのデータは**ファイル**という単位で扱われる．ファイルには，`hoge`, `fuga.c`, `hena.tar.gz` のような**ファイル名**が付く．ファイル名のうちピリオドより後ろの部分を**拡張子**と呼び，ファイルの種類等を表す特別な意味を持たせているファイルシステムもある (☆ 5)．ファイル名に使える文字の種類やファイル名の長さなどは，ファイルシステムによって規定されている．

応用プログラムからは，ファイルは一定の論理的な構造を持っているように見える．この構造を**ファイル編成**という．次の 2 つが基本的なものである．

ファイル: file

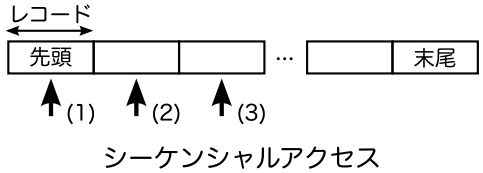
☆ 5) 例えば，MS-DOS や昔の Windows で採用されていた FAT というファイルシステム（の基本形）では，ファイル名は (8 文字).(拡張子 3 文字) の形式だった．

順編成 一つのファイルは一定サイズのデータ（**レコード**という）．が一列に並んでできている．先頭のレコードから順にアクセスすること（**シーケンシャルアクセス**）しかできない（後ろの方のレコードだけにアクセスしたくとも先頭から順にたどっていかねばならない）．

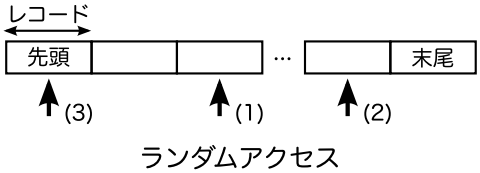
シーケンシャルアクセス: 順次アクセス，sequential access.

直接編成 レコードが並んでファイルができているのは順編成と同じであるが，ファイル中の位置を指定して特定のレコードのみにアクセスすること（**ランダムアクセス**）ができるような仕組みを備えたもの (☆ 6)．

☆ 6) ランダムアクセス: random access. 磁気テープを用いる補助記憶装置のようにランダムアクセス不可能なものもある．



シーケンシャルアクセス



ランダムアクセス

汎用コンピュータ (☆ 7) 向け OS で用いられるファイルシステムでは，これらに加えてもっと複雑なファイル編成を採用しているものが多い．一方，PC 向け OS ではより単純なものが使用される．例えば，UNIX 系 OS で一般に用いられるファイルシステムでは，1 バイト単位でアクセス可能（レコードの大きさが 1 バイトともみなせる）で，ごく単純なランダムアクセスのみが可能となっている (☆ 8)．

ファイルに対しては，一般に次のような操作が行える： ファイルの生成／削除，ファイルのオープン／クローズ，データの読み込み／書き込み／追加書き込み，ファイル名の変更．応用プログラムは，OS の API を介して（API 関数の呼び出し／システムコールによって）これらのファイル操作を行う．

☆ 7) 汎用コンピュータ: 企業等で業務に使われるような大型のコンピュータ．メインフレームという言い方もある．

☆ 8) ファイル中のどの位置で読み書きするかをファイルの先頭からのバイト数で指定することができる．C 言語だと `fseek()` を使う．

★ 1.2 ディレクトリ

一般的なファイルシステムでは，**ディレクトリ** (☆ 9) の階層を用いてファイルを整理する．この階層は，下図の例に示すように木構造の一種とみなすことができる．そのため，階層の最上位のディレクトリを**ルートディレクトリ**と呼ぶ．

ファイルシステム中のファイルは，`/home/t070000/hoge.txt` のような「ディレクトリ名のならび+ファイル名」で識別される．これを**パス**という (☆ 10)．パスとは「経路」という意味であり,「ルートディレクトリから `home` に行って `t070000` に行ってその下の `hoge.txt`」というようにそのファイルへのたどり着き方を表している．パスは，上記のようにルートディレクトリを起点として表現することもできるし，適当なディレクトリを起点として表現することもできる．前者を**絶対パス**，後者を**相対パス**という．例えば，絶対パス `/usr/bin/cc` のファイルは，`/usr` を起点とする相対パスでは `bin/cc` と表される．

通常，プロセスは，それが用いる相対パスの起点をどこかに定めた上で実行される．そのディレクトリを**カレントディレクトリ**という．下図の例でカレントディレクトリを `aprog2008` として `a.out` というプロセスを実行 (☆ 11) したとすると，このプロセスでの `hoge.txt` という相対パスは，`/home/t070000/aprog2008/hoge.txt` を指すことになる．UNIX 系 OS では，カレントディレクトリとその上のディレクトリを「`.`」と「`..`」で表し，相対パスに用いることができる (☆ 12)．カレントディレクトリは，シェル上で `cd` コマンドを用いて移動させることができる．また，`pwd` コマンドでカレントディレクトリを表示させることができる．

☆ 9) ディレクトリ: directory. OS ／ファイルシステムによってはフォルダ (folder) とも．ルート: root, 根．パス: path
木: スタックやキューと同じく代表的なデータ構造の一つ．

☆ 10) ここで説明しているパスの表記例は UNIX 系 OS で用いられるものである．先頭の “/” はルートディレクトリを表し，それ以外の “/” はディレクトリ名の区切りを表す．Windows では “\” を用いる (ややこしい事情のため，日本語環境では “¥” になったりする)．
current a. 現在の．カレントディレクトリは「ワーキングディレクトリ」ともいう．

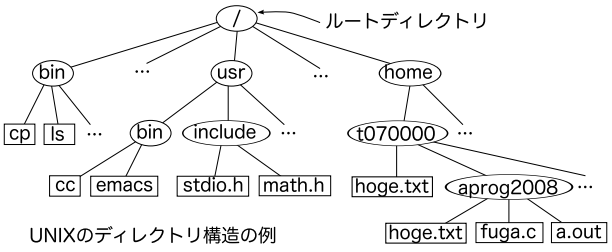
☆ 11) 例えば，シェル上で `cd` コマンドで `aprog2008` に移動してから `./a.out` と実行．

☆ 12) ↑の場合，
相対パス `../hoge.txt` は `/home/t070000/hoge.txt` という絶対パスを表す．

Q1. 右図の例で，次のファイルの絶対パスを示せ．
`ls`, `stdio.h`, `fuga.c`

Q2. 次のディレクトリを起点とする `/bin/ls` の相対パスを示せ．`/bin`, `/usr`, `/usr/bin`

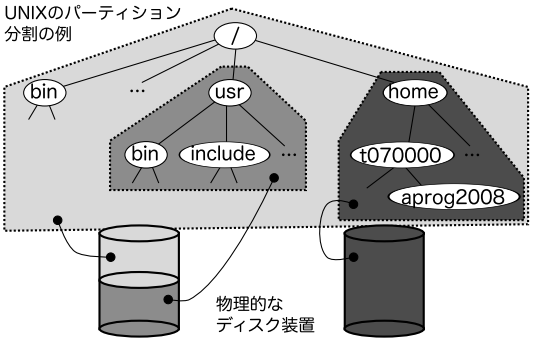
Q3. カレントディレクトリが `/usr/include` であるときに次の `cd` コマンドを実行すると，カレントディレクトリはそれぞれどこに移るか．
`cd ../bin`, `cd ../../bin`, `cd ../../usr/bin`



UNIXのディレクトリ構造の例

★1.3 パーティション／ボリューム

一般に，一つのファイルシステムは複数の**パーティション**と呼ばれる区画から構成される．ディスク装置の記憶領域全体を一つのパーティションとすることもあれば，一つのディスクを複数のパーティションに分割することもある（複数のディスクをまとめて一つのパーティションを構成できるファイルシステムもある）．パーティションは，論理的なディスク装置とみなせる．



右図に示すように，UNIX のファイルシステムでは，全てのパーティションを接ぎ木して一つの木を構成する（これを「マウントする」という）ようになっている (☆ 13)．これに対して，Windows で用いられるファイルシステムでは，パーティション毎に C,D, 等のアルファベット（ドライブレター）が割り当てられて，それぞれ独立したディレクトリ木を構成するようになっている．

パーティション: partition, ボリューム (volume) とも．

☆ 13) UNIX 系 OS では，`mount` や `df` といったコマンドでマウント状況を確認できる．

★1.4 ファイルの属性

多くのファイルシステムでは，ファイル名の他にも様々な属性をファイルに付与することができる．例えば，UNIX 系 OS で一般的なファイルシステムでは，

- そのファイルの持ち主のユーザ ID とグループ ID(☆ 14)
- モード（下記参照）
- 最終更新日時と最終アクセス日時
- 長さ（ファイルの大きさ）

等の属性が存在する（↓でわかるようにディレクトリにも属性がある）．

☆ 14) ユーザは 1 つ以上の「グループ」に所属することになっている．自分の所属は `groups` コマンドでわかる．

```
                ユーザ名 グループ名  サイズ     更新日時     名前
$ ls -l              ↓       ↓       ↓          ↓          ↓
total 64
drwxr-x---    2 takataka hogehoge       80 Dec 18 15:26 hoge/
 ← ディレクトリを表す
-rwxr-xr-x    1 takataka hogehoge    13278 Jan  5 15:05 sammokumain*
 ← モード：左から3文字ずつ，持ち主，グループメンバ，その他のユーザそれぞれが，
   読み(r),書き(w),実行(x)が可能かどうかを表す．この場合，持ち主はどれも可，
   その他は読んで実行できるが修正削除はできない．

-rw-r-----    1 takataka hogehoge     1257 Jan  5 15:04 sammokumain.c
-rw-r--r--    1 takataka hogehoge     2136 Jan  5 15:05 sammokumain.o
$ chmod g+w sammoku.c  ← グループメンバに書き込み許可を出す
$ ls -l sammokumain.c
-rw-rw----    1 takataka hogehoge     1257 Jan  5 15:04 sammokumain.c
```

これらは，上記のように，`ls` コマンドをオプション`-l`(☆ 15) 付きで実行すれば確認できる．属性の中には，そのファイルに対して誰にどのような操作を許可するかを表す**モード**という情報もあり，これによってファイルを保護する仕組みになっている．モードは，`chmod` というコマンドで変更することができる（詳細は `chmod` のマニュアル（`man chmod` を実行すれば表示される）参照 (☆ 16)）．

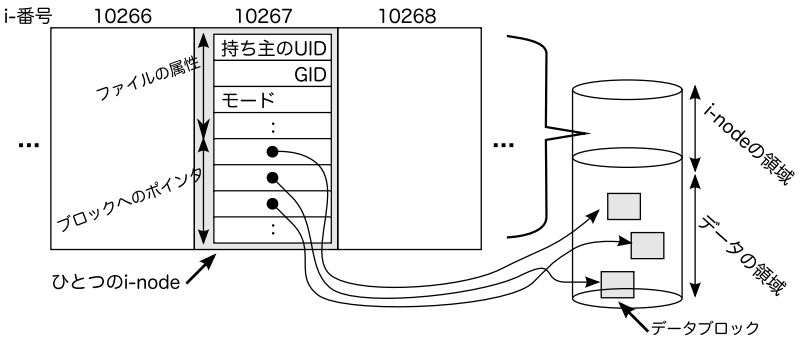
☆ 15) `l` は `L` の小文字．オプション次第でアクセス日時や i-番号（後述）も表示できる．`man ls` 参照．

☆ 16) ファイルの持ち主やグループを変えるコマンドもある (`chown`, `chgrp`).

★1.5 i-node

ファイル名，ファイルの属性，ディレクトリ構造などがファイルシステム上でどのように表現されているかを考えよう．ここでは，UNIX 系 OS のファイルシステムで一般的な仕組みを解説する．

UNIX で一般的なファイルシステムでは，ファイルの内容は，ディスク中の連続した領域としてではなく，**ブロック**と呼ばれる一定サイズ（多くの場合 512 バイト）の領域に分割して扱われる．そのため，一つのファイルを構成するブロックがディスク中のあちこちに存在することになる (☆ 17)．したがって，ブロックがディスク中のどこにあるかを表す情報を何らかの方法で管理する必要がある．



このような情報は，ファイルの属性とともに，**i-node** と呼ばれる構造体に格納される (☆ 18)．ひとつの i-node がひとつのファイルに対応し，それぞれに **i-番号**という一意な番号が振られるようになっている（上図参照）．

i-node は，ディレクトリとその階層構造を表すのにも用いられる．ファイルと同様に，ひとつの i-node がひとつのディレクトリに対応し，その中に属性の情報とデータブロックへのポインタが格納される．そして，データブロックの中に，そのディレクトリ中に存在するファイルやディレクトリの名前と i-番号の対応表が格納される（下図左参照）．

このような仕組みであることから，例えば `/hoge/fuga.c` という絶対パスをもつファイルを読み書きする際には，以下に示すようにルートディレクトリの i-node から順にたどっていくことになる (☆ 19)．

1. ルートディレクトリの i-node(☆ 20) を参照し，それが指すデータブロックの中から `hoge` という名前に対応した i-番号を探す（下図では 5963）．
2. 上記の i-番号をもつ i-node を参照し，それが指すデータブロックの中から `fuga.c` という名前に対応した i-番号を探す（下図では 4649）．
3. 上記の i-番号をもつ i-node を参照し，それが指すデータブロックにアクセスする．

また，`mv` コマンドによるパスの変更は，ディレクトリ中のデータの書き換えによって実現されている．したがって，ファイルをあるディレクトリから別の場所へ移動させても，データを実際にディスク中で移動させる必要はない (☆ 21)．同様に，`rm` コマンドでファイルを消去する操作は，そのファイルの名前と i-番号をディレクトリのデータから消去することで実現されている (☆ 22)．

☆ 17) そのかわり，ファイルサイズを変更する際にはブロックを追加したり削除したりするだけで済む（複数のファイルをディスクの先頭から順に詰めて書き込むような仕組みだったらおおごと）．

☆ 18) i-node の情報は，すばやくアクセスできるように，ディスクの先頭の領域にまとめて書き込まれる．

☆ 19) 相対パスで指定した場合も，カレントディレクトリと相対パスから絶対パスを求めれば後は同じ．

☆ 20) ルートディレクトリの i-番号は決まっている（2 であることが多い）．また，ルートディレクトリの親ディレクトリはルートディレクトリ自身と定められている．

☆ 21) ファイルシステムを越えた移動の場合は実際にデータを移動させねばならないこの限りではない．

☆ 22) したがって，そのファイルの i-node とデータブロックの内容は，やがて使い回されるまでそのまま放置される．そのため，特殊なソフトウェアを用いると，運が良ければ一度消去したファイルを復活させられることがある．

★ ディレクトリのブロックの内容
(/ のi-番号が 2 で，/hoge のi-番号が 5963 の場合)

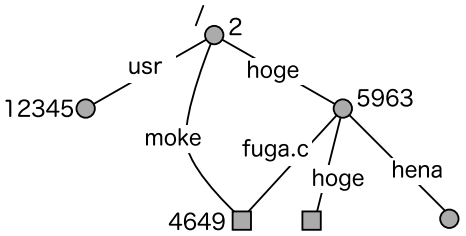
#2のデータブロック

名前	i-番号
.	2
..	2
hoge	5963
usr	12345
moke	4649
:	:

#5963のデータブロック

名前	i-番号
.	5963
..	2
fuga.c	4649
hena	67417
hoge	999
:	:

★ 左の構造を図示したもの
（●はディレクトリ，■はファイル）



ファイル /hoge/fuga.c と /moke の実体は同一

★ 1.6 ［発展］ リンク

上述のことから分かるように，実はファイルやディレクトリの名前は i-node には格納されておらず，ディレクトリのデータとして i-node と対応づけられているだけである．このことは，上図右が示すように，これらの名前はファイルやディレクトリそのもの（図の●や■）ではなく，それらを結ぶリンクに付いていることを意味している．そのため，図のように，/hoge/fuga.c と/moke が同じファイルを指すようにする（別名をつける）こともできる (☆ 23)．

ln コマンドを用いると，実際にこのような操作（「**リンク**する」または「リンクを張る」という）を行える．上記のことを実現するには，/hoge/fuga.c が既に存在するときに「$ ln /hoge/fuga.c /moke」とすればよい (☆ 24)．

上記のようなやり方でファイルに別名をつける操作のことを，より詳しくは**ハードリンク**といい，別種の実現方法である**ソフトリンク**と区別することがある．

☆ 23) 上図左に示すように，/hoge/fuga.c の i-番号 4649 を moke という名前で/中の対応表に載せてしまえばよい．

☆ 24) $ rm /hoge/fuga.c してもこの場合 i-番号 4649 のファイルそのものはなくならない．

ハードリンク 上述のようなもの．1 つの実体に複数のハードリンクがあってもそれらは対等である（どれか 1 つリンクを削除しても実体には影響しない）．

ソフトリンク 実体へのリンクではなく，リンク先へのパスの情報のみを保持するもの．例えば，/hoge/fuga.c をリンク先（オリジナル）として，/moke にはそれへのパスの情報のみを保持する．後者へのアクセスは，そこに記された情報を手がかりにオリジナルのファイルへのアクセスとして扱われる．オリジナルを削除・移動するとソフトリンクはたどれなくなってしまう．

UNIX 系 OS ではソフトリンクのことをシンボリックリンクと呼ぶ (☆ 25)．Windows の「ショートカット」はソフトリンクの一種と見なせる．

☆ 25) ln コマンドに-s オプションをつけるとシンボリックリンクを張ることになる．ディレクトリへのハードリンクは（原理的には可能だが混乱を避けるために）できないようになっているので，ディレクトリに別名をつけるにはこちらを用いる．

★ 1.7 ［発展］ ネットワークを介したファイルの共有

龍谷大学の計算機室では，どの PC を使っても自分のホームディレクトリにアクセスできる．しかし，よく考えたらこれは不思議なことである．これは，ネットワークを介して遠隔地のコンピュータ（ファイルサーバ等と呼ばれる）上のディスク装置を目の前のコンピュータが共有してアクセスできるようにする仕組みによって実現されている．あ，もう余白がない… (;_;