Raid

目的

- ハードディスクの信頼性が問題
  - →1個が壊れてもデータは安全にしたい。
- 連続運転の為にはホットスワップが必要

Raid技術の誕生

- “A Case for Redundant Arrays of Inexpensive Disks(RAID)” David A Patterson et al., 1988
  - **カリフォルニア大学バークレー校の**David A.Patterson**氏、**Garth Gibson**氏、**Randy Katz**氏の共同論文「安価なドライブを組み合わせることで冗長性を持たせる仕組み」による**
  - **安価なディスクをたくさん使う**
  - **いか**RAID**と略される**

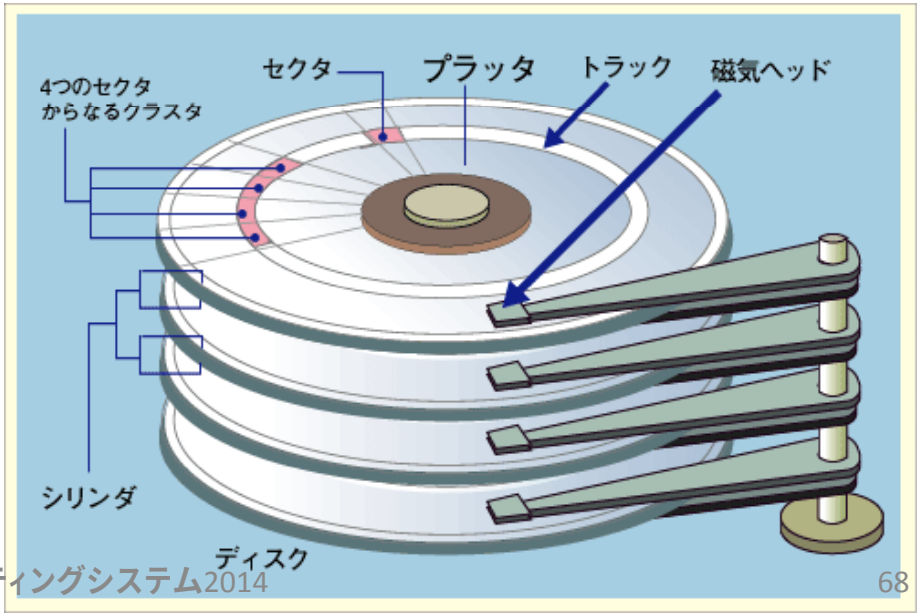
- **ソフトウェア**RAID
  - **無料**
- RAID**カード**
  - **数万円～１０万円**
- **内臓**RAID**ユニット**
  - **数万円**
- **外付け**RAID**ユニット**
  - **数十万円～数百万円**
- RAID**サーバ** (NAS**サーバ**)
  - **数十万円～数千万円**

- Redundant Arrays of Independent Disks
  - RAID Advisory Board **での定義**

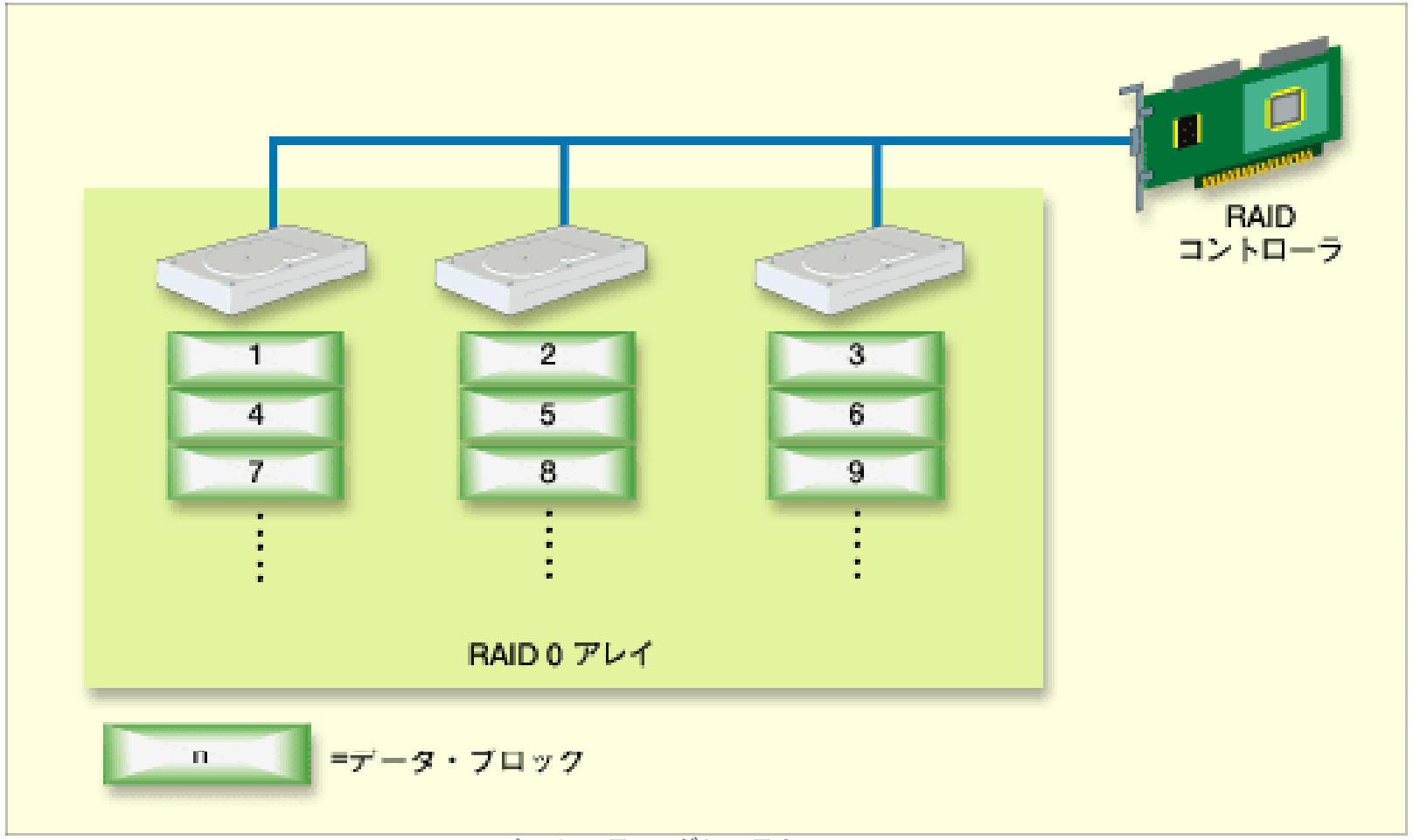
- **レベル**0　**ストライピング**
- **レベル**1　**ミラーリング**
- **レベル**2　**分散ハミングコード**
- **レベル**3　**バイト分割固定パリティ**
- **レベル**4　**ブロック分割固定パリティ**
- **レベル**5　**分散パリティ**
- **レベル**6　2**重分散パリティ**

磁気ディスクの構造

- とにかく読み書きに時間がかかる。
- 全部のディスクが同時に回る。
  - メモリの探索速度はディスクの10万倍〜100万倍高速。
  - メモリの転送速度もディスクのざっと100倍高速。



Raid0



Raid0 ストライピング

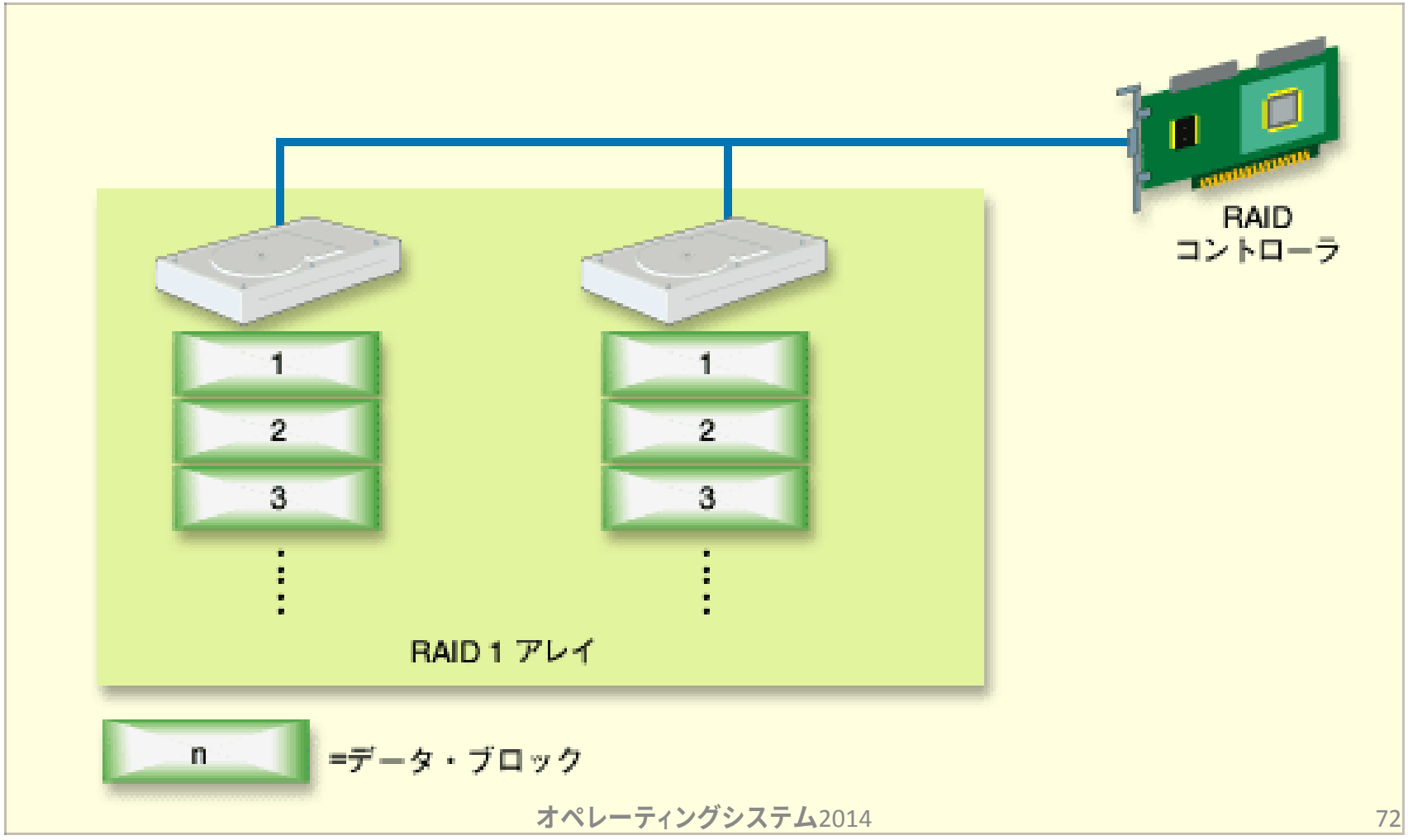
- データ･ブロック1／2／3や4／5／6、7／8／9は、もともと連続しているデータである。
- RAID 0では、ディスクの台数に合わせてデータを分割して、各ディスクに格納する。
- 例えばデータ･ブロック1／2／3の組を読み出す場合、各ディスクに並行してアクセスすることで、ほぼ同時に1／2／3それぞれのデータ･ブロックを読み出すことが可能。

- 高速化技術

Raid1　ミラーリング

- RAID 1は、RAIDレベルの中で、最も単純な手法でディスクの耐障害性を高めている。その手法とは、同一のデータを複数のディスクに書き込み、一方のディスクが故障しても、他方で処理を続行できるようにする。

Raid1ミラーリング

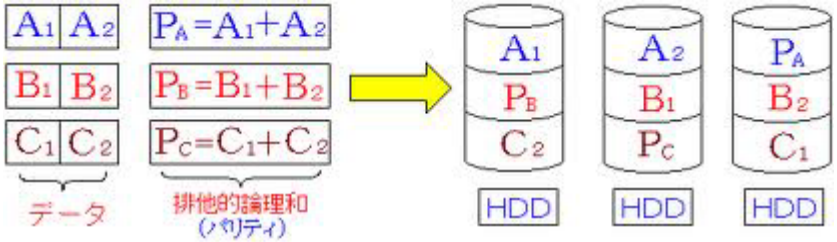


Raid1 稼働率

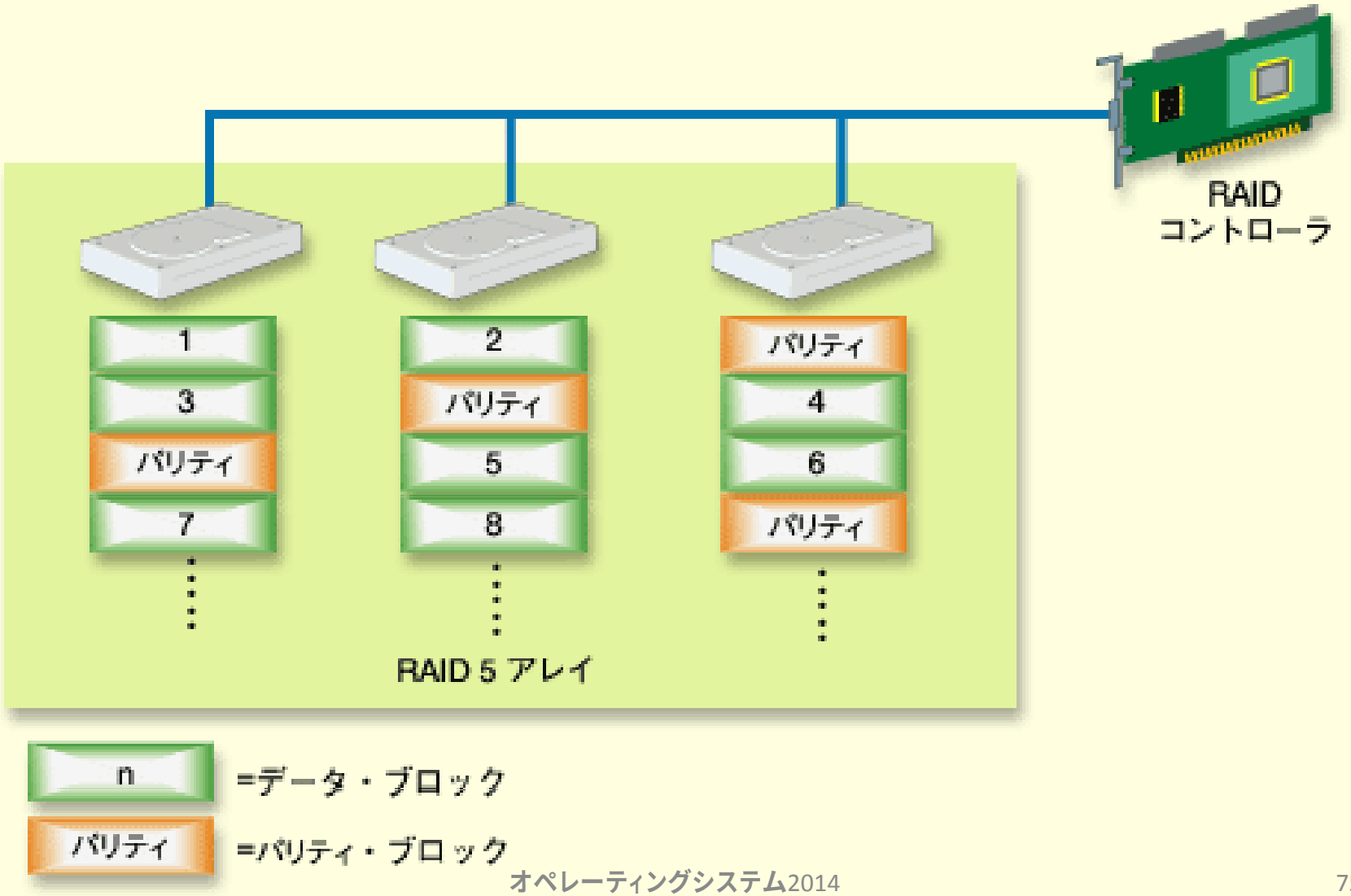
- RAID 1では、同一のデータを2台以上のディスクに書き込むため、ディスク容量の利用効率は50%以下になってしまうというデメリットがある（2台のディスクの 容量が異なると、利用効率は50%よりさらに下がる）。
- 利用効率=保存可能データ量/利用ディスク
  =100%n/2n=50%
- 例えば1Tbytesのデータを記録するには、1Tbytes×2＝2Tbytes分の容量のディ スクが必要になる。

Raid5

- RAID 5は、耐障害性の向上と高速化、大容量化のすべてを実現できるRAID技術。
- **分散データ･ガーディング**とも呼ばれる。
- RAID 5では、ディスクの故障時に記録データを修復するために「パリティ」と呼ばれる冗長コードを、全ディスクに分散して保存するの



Raid5


RAID
コントローラ
1
3
パリティ
7
2
パリティ
5
8
パリティ
4
6
パリティ
RAID 5 アレイ
n
=データ・ブロック
パリティ
=パリティ・ブロック

RAID 5の動作原理

- データを分割して各ディスクに格納するという原理はRAID 0（ストライピング）と同じだ。異なるのは、データ・ブロックの組（上図でいえば1／2や3／4、5／6）ごとにパリティが生成される点である。たとえ1 台のディスクが壊れても、残りのディスクに格納されたデータとパリティから、失われたデータを復活させることができる。

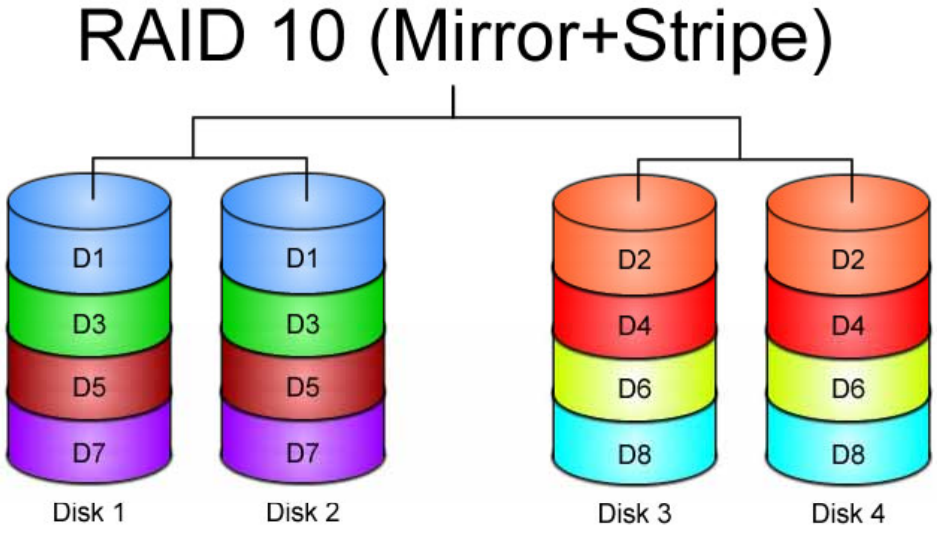
Raid5の利用効率

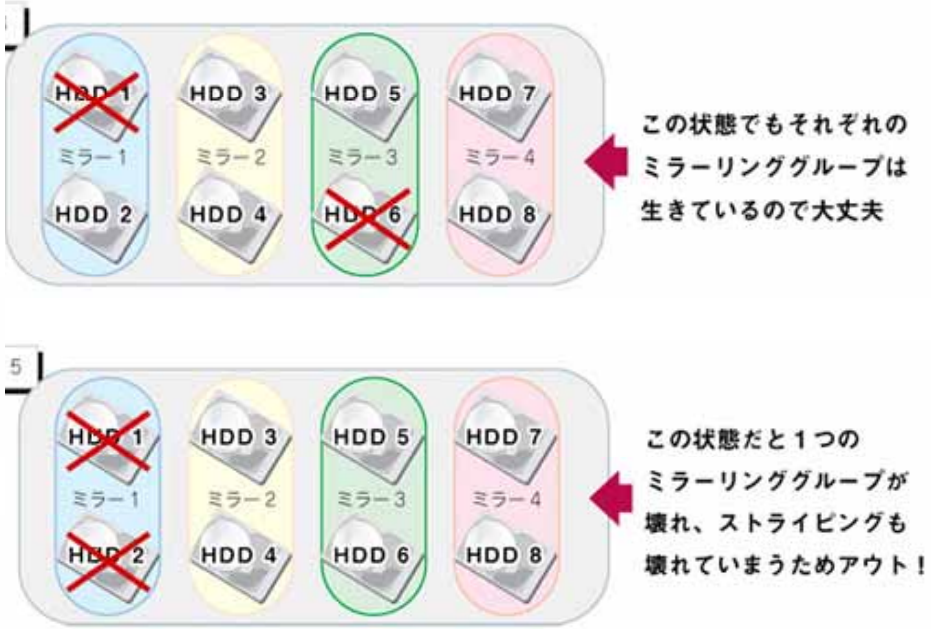
- **パリティの保存に必要なのは、全ディスク台数に関係なくディスク1台分の容量である。従ってディスク台数が多いほど容量の利用効率も向上する。**RAID 1 (**ミラーリング**) **と比較した場合、この利用効率の高さが**RAID 5**のメリットの**1**つとされる。**

- **ディスク容量の利用効率**
  - 100 * (n-1)/n %

Raid10 Raid1+0

- **ミラーリンググループをストライピング**
- **冗長性と高速性**





その他(覚えなくて良い)

- **RAID 2**
- RAID 2では、エラーを修復するための冗長コードを元のデータとともに、複数のディスクにまたがって記録する。特徴は、冗長コードが多ビットのECC (Error Correcting Code；誤り訂正符号) であることと、データを配分するときの単位サイズがブロック (セクタ) 単位ではなくbitまたはbyte単位。
- 多ビットのECCによる冗長コードは、元のデータに対してそのサイズが大きくなりがちで、容量面でのオーバーヘッドが大きいというデメリットがあ る。例えば、代表的なECCの1つであるハミング符号を用いると、元のデータを2台のディスクに分散するには、冗長コードだけのために3台のディスクが必 要になってしまう。元のデータより冗長コードの容量を小さくするには、元のデータを格納するディスクを4台以上にしなければならない。

その他 (覚えなくて良い)

- RAID 3
  - 元のデータに冗長コードを加えて複数のディスクに記録。
  - 冗長コードには、RAID 5と同じパリティを採用する。
  - RAID 5と大きく異なるのは、各ディスクにデータを配分する際の単位サイズが、ブロック (セクタ) 単位ではなくbitまたはbyte単位
  - パリティが特定のディスクに保存される (全ディスクには分散されない)
  - 高速化のためには、複数のディスクをまったく同時に読み書きするための同期機能が必要

- RAID 4
  - RAID 4は、RAID 5と同様に、元のデータからパリティを生成して、ブロック単位で複数のディスクに記録する、という点でよく似ている。異なるのは、パリティを全ディスクに 分散するのではなく、
  - 特定のディスクだけに格納する点だ。
    - 元のデータとパリティそれぞれを格納するディスクが別々に分かれている

その他（覚えなくてよい）

- **RAID 6**
  - RAID 6は、RAID 5の改良版といえる技術
  - 1つのデータ・ブロックにつき2つのパリティを生成
  - 同時に2台のハードディスクが故障しても、元のデータを修復可能
  - パリティが増えた分、その計算や書き込みのオーバーヘッドも増加する
  - 特に書き込みの性能は高くない。
  - パリティ用に2台分のディスク容量を必要とするため、ディスクの利用効率はRAID 5より下がる。

パス名（絶対・相対パス）

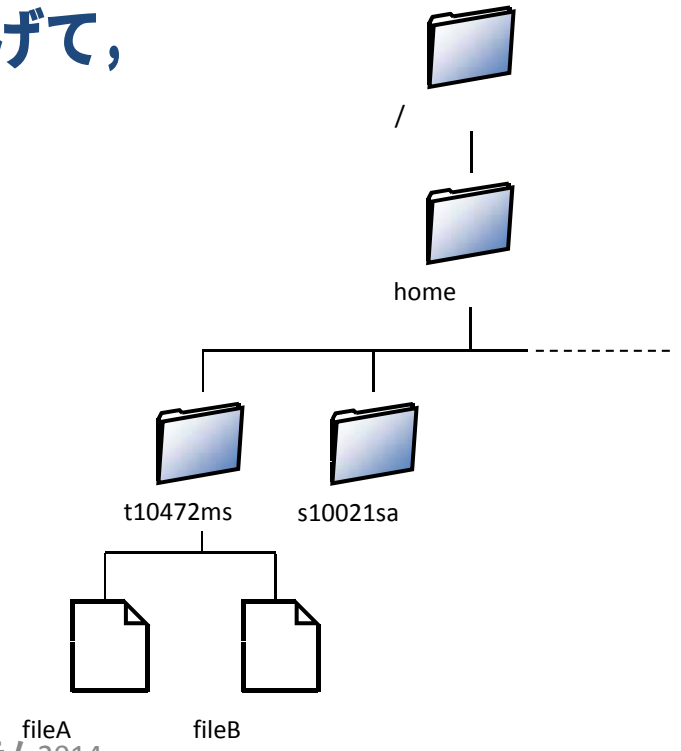
パス名 (絶対パスと相対パス)

- ファイルやディレクトリにアクセスするために，ファイルやディレクトリの位置 (パス名) を示す必要がある
- パス名の指定方法は以下の2種類

- 絶対パス
  - ルートディレクトリを基点として絶対的な位置を指定する
  - 例：住所は絶対パス「東京都足立区千住旭町5」
- 相対パス
  - あるディレクトリを基点にした相対的な位置を指定する
  - 場合によっては，絶対パスより短いパス名で指定できる
  - 例：田中君の家は「私の家の右隣」

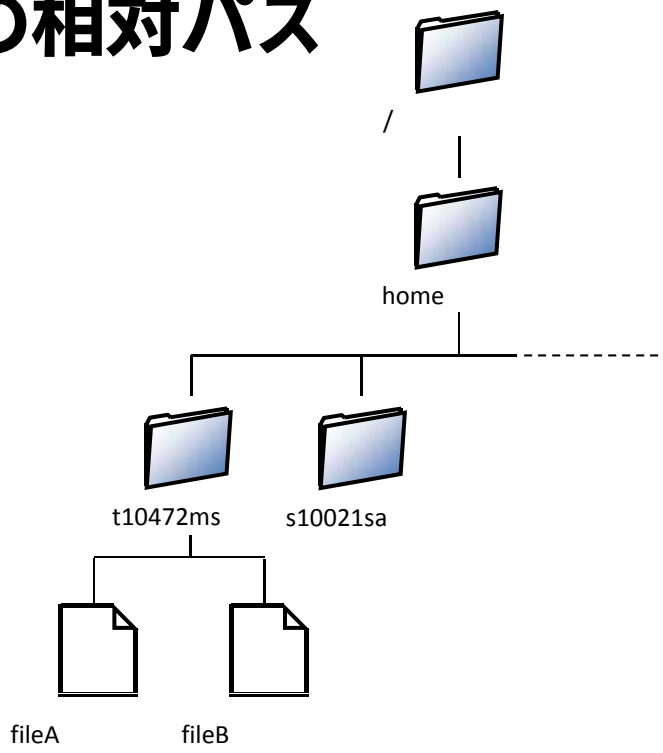
絶対パス

- fileAの絶対パス
  - 日本語だと“ルートディレクトリの中の，homeディレクトリの中のt10472msディレクトリの中のfileA”
  - 区切りを「/（スラッシュ）」で繋げて，“/home/t10472ms/fileA”



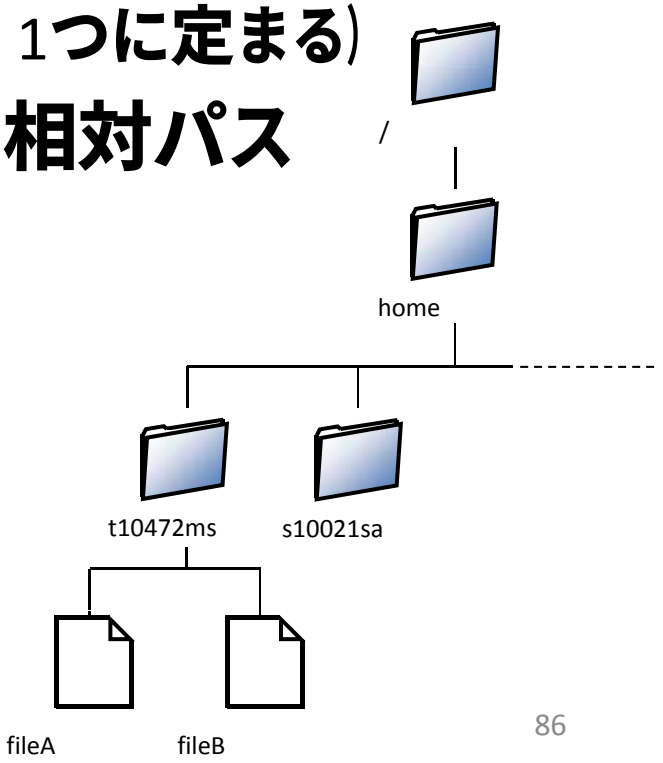
相対パス1

- t10472msを基点にした時のfileBの相対パス
  - **“fileB”**
- homeを基点にした時のfileBの相対パス
  - **“t10472ms/fileB”**



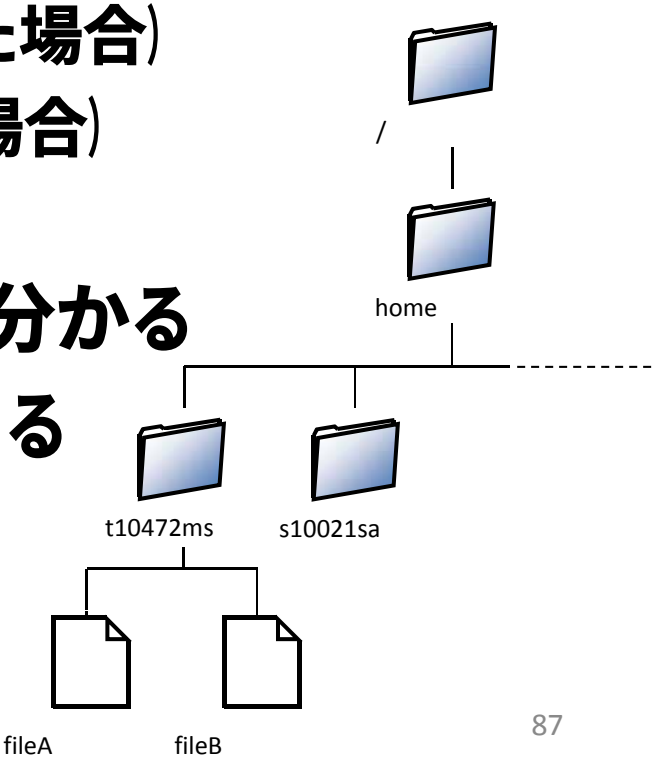
相対パス2(重要)

- **基点から見た親ディレクトリを指定するには，“.. (ピリオド2つ) ”の記号を使う**
- t10472msを基点にした時のhomeの相対パス
  - **“..” (親ディレクトリは1つだけなので，1つに定まる)**
- s10021saを基点にした時のfileBの相対パス
  - **“../t10472ms/fileB”**



相対パス3

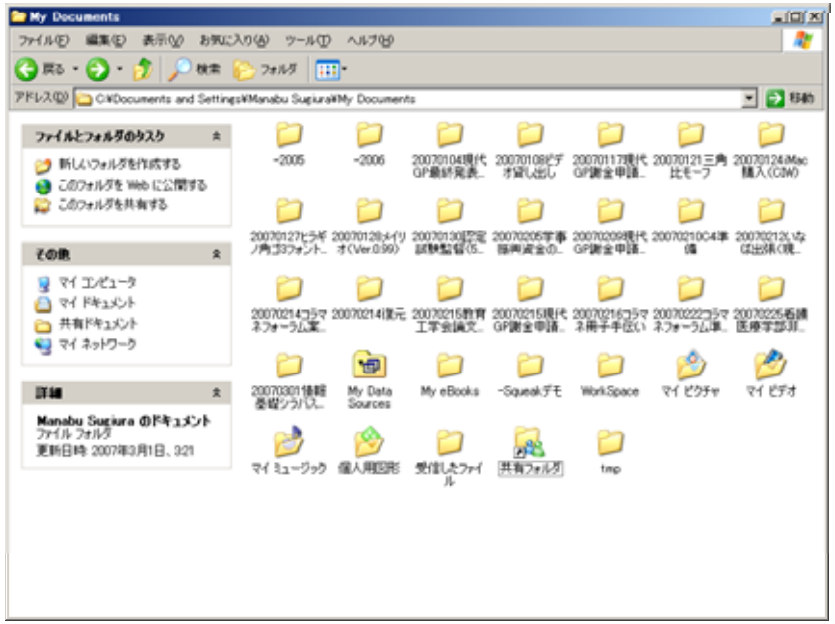
- 基点のディレクトリを示すには，“. (ピリオド1つ) ”の記号を使う
- t10472msを基点にした時のfileAの相対パス
  - “./fileA” (基点ディレクトリを明示した場合)
  - “fileA” (基点ディレクトリを省略した場合)
- 基点ディレクトリを明示すると，相対パスによる指定であることが分かる
- パス名が読みやすくなる場合がある



ファイルとディレクトリの操作

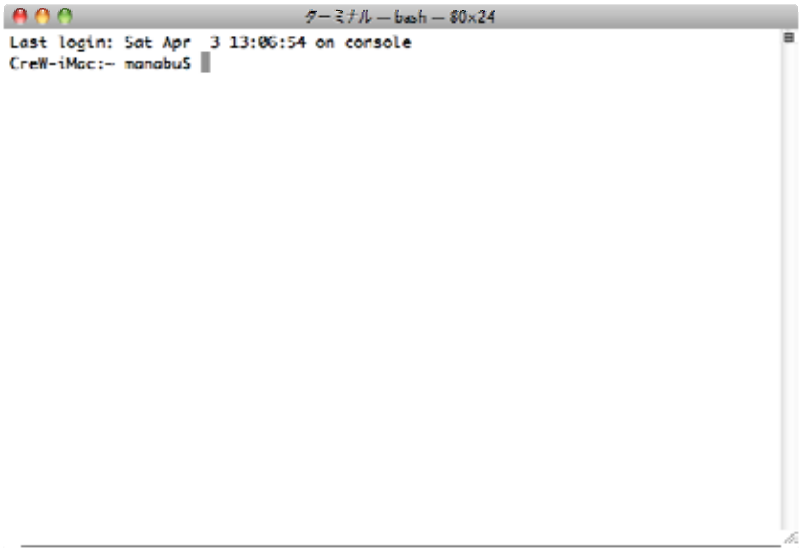
ファイルの操作方法1
ファイルマネージャーを使う

MacのFinder

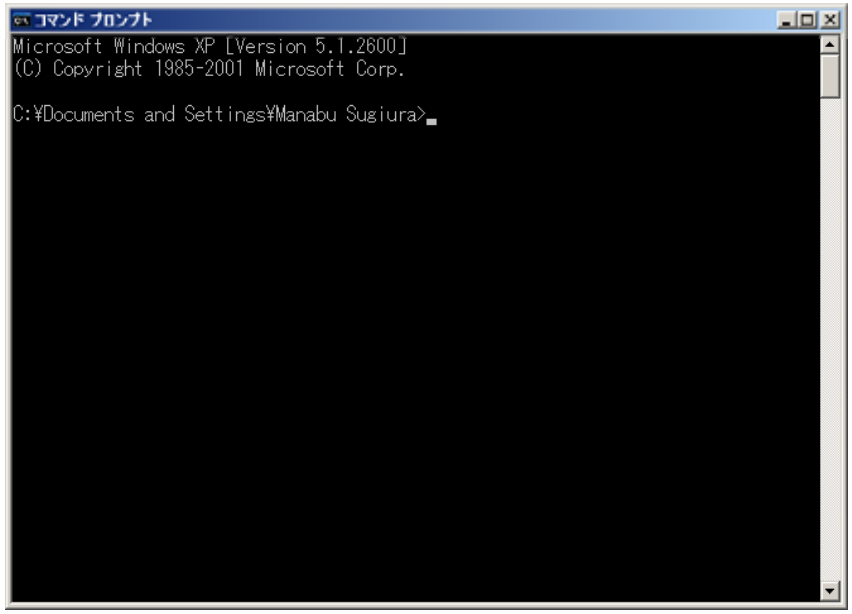


WindowsのExplorer

ファイルの操作方法2
コマンド操作でファイルを管理する



Macのターミナル



Windowsのコマンドプロンプト

GUIとCUI

- Graphical User Interface (GUI)
  - 画面表示にアイコンやメニューを用い，操作の大半をマウスなどのポインティングデバイスによって行なう
  - Finder (Mac) やExplorer (Windows) はGUIを備えたファイルマネージャ (ファイル管理機能をもつソフトウェア)
  - 直感的に操作ができる

- Character User Interface (CUI)
  - すべての操作をキーボードからコマンドと呼ばれる命令を用いて行なう
  - ターミナル (Mac) やコマンドプロンプト (Windows) を使うとCUIを使ってコンピュータを操作できる
  - 効率よく命令を記述でき，慣れれば素早く操作を行える

ターミナル

- コマンド操作を行うためには，ターミナルというプログラムを使う

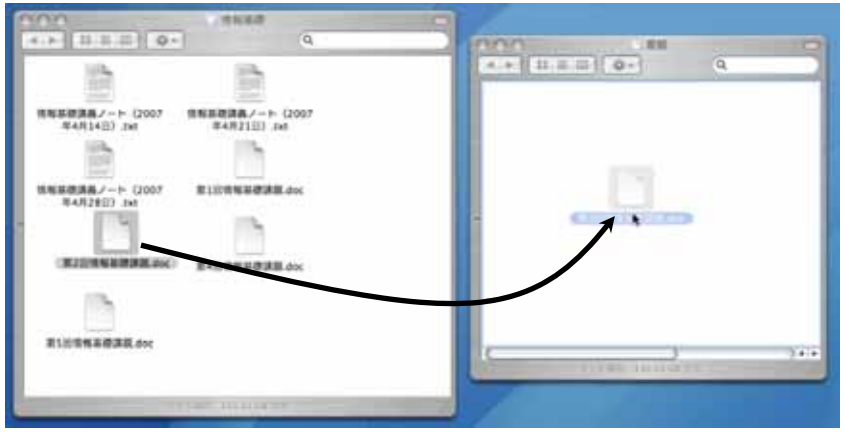
- 起動方法
  - 初期設定ではDockに登録されている
  - Dockにない場合
    - Finderを起動する
    - サイドバーのアプリケーションを選択する
    - ユーティリティフォルダ中の，ターミナルをクリック

コマンド

- コンピュータに与える命令のこと
- CUIのターミナルでは（マウスによるボタン操作ではなく），文字で命令を伝える

- 例：ファイルの移動

Finder（GUI）だとマウスで命令



ターミナル（CUI）だと文字で命令

```
% mv 第2回情報基礎課題.doc /Users/
```

プロンプト

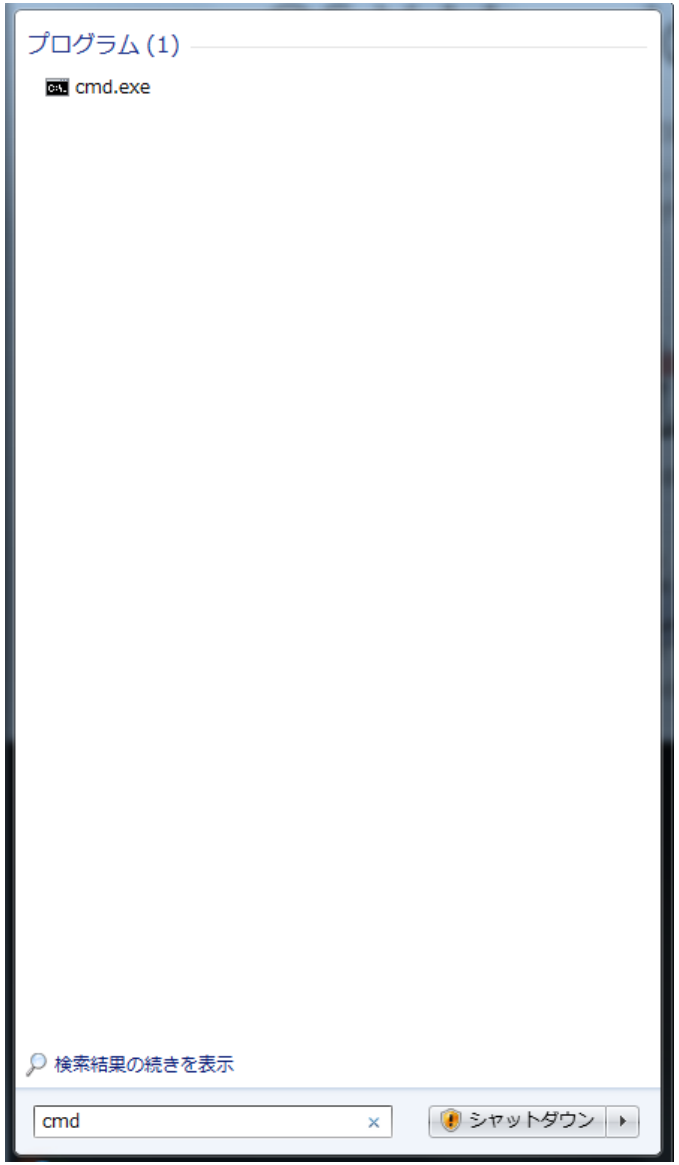
- ターミナルを起動すると，%マークが現れる
- これをプロンプトと呼び，コンピュータがコマンドによる指示を待っている印 (しるし)
- コマンドを入力したら，エンターキーを押すと命令が実行される

```
% コマンド ↵
```

このスライドではエンターキーを押すタイミングを↵で表現しています

コマンドプロンプト

cmd.exe



なぜコマンド操作を学習するか

- サーバはコマンドで操作することが多い
  - プロバイダから提供されているWebサーバを設定する
  - 所属する研究室・会社のサーバの管理をする
- 効率よくコンピュータに仕事を指示することができる
  - シェルスクリプト
  - ワイルドカード

日付とカレンダーの表示

- **日付の表示**：dateコマンド

```
% date ↵
2010年04月02日（金）10時49分52秒 JST
```

- **カレンダーの表示**：calコマンド

```
% cal ↵
 2010年  4月
日 月 火 水 木 金 土
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

ファイル･ディレクトリの操作の コマンド一覧

- pwd → カレントディレクトリの絶対パスを表示
- ls → ディレクトリの内容を見る
- cd → カレントディレクトリの移動
- less → ファイルの内容を見る
- mkdir → 新しいディレクトリを作る
- cp → ファイルのコピーを作る
- mv → ファイルの移動･ファイル名の変更
- rm → ファイルの削除
- rmdir → ディレクトリの削除

カレントディレクトリ

- コマンドによってファイルやディレクトリを操作する場合，相対パスでファイルやディレクトリを指定する方が便利
- 現在の作業ディレクトリのことをカレントディレクトリ (ワーキングディレクトリ) という
- カレントディレクトリからの相対パスでファイルを指定することができる
- ターミナルを起動した直後のカレントディレクトリは「ホームディレクトリ/CNSiMac」になる

カレントディレクトリの表示

- pwd（print working directoryの略）コマンド
  - カレントディレクトリの絶対パスを表示する

```
% pwd ↵
/a/fs0102a/t10472ms
```

ファイルサーバは何台かのコンピュータで分担してホームディレクトリを保管しているので，本当のホームディレクトリの絶対パス名は /a/fs0102a/t10472msのようにファイルサーバの番号とログイン名を組み合わせたものになっています
誰のホームディレクトリがどのファイルサーバにあるかを覚えるのは大変なので，/a/fs0102a/の部分をまとめて，'/home' と表わします

ディレクトリの内容を見る1

- ls（listの略）コマンド
  - カレントディレクトリにあるファイルとディレクトリの一覧を表示する

```
% ls ↵
Desktop         Maildir         XPAppCNS
Wnn             XPDataCNS
```

ディレクトリの内容を見る2

- -a オプション
  - .emacsのように先頭がドットで始まるファイルは，ソフトの設定に使うファイルなので普通は表示されない
  - ls コマンドに –a オプションを付けると表示できる
  - ls のあとに1つ空白をあけてからオプションを入力する
  - オプションをつけることで，コマンドの機能を拡張できる

```
% ls -a ↵
.                       .gnome2                 .w3m
..                      .gnome2_private         .winman
.ICEauthority           .gstreamer-0.8          .xsession-errors
.cshrc                  .gtkrc-1.2-gnome2       Desktop
.emacs                  .metacity               Maildir
.emacs.d                .mh_profile             Wnn
.folders                .mozilla                XPAppCNS
.fonts.cache-1          .nautilus               XPDataCNS
```

ディレクトリの内容を見る3

- ディレクトリのパス名を引数（ひきすう）として指定
  - カレントディレクトリ以外のディレクトリを見たいときは，そのディレクトリのパス名を ls の後につける
  - コマンドの後に1つ空白をあけてから付け加えるものをこのコマンドの引数（ひきすう）と言う

```
% ls Maildir↵
courierimaphieracl      courierimapuiddb        tmp
courierimapkeywords     cur
courierimapsubscribed   new
```

【演習 Mac】
lsコマンドを極めよう

- **自分のホームディレクトリにあるファイルのうち最も新しいファイルを見つけてみよう**
  - **ファイルを新しい順に表示するオプションは** -t
- **実験してみよう**
  - ls **の引数に存在しないディレクトリ名を指定してみる**
  - ls -l **の引数にディレクトリでは無く，ファイルを指定してみる**
- ls **コマンドのその他の機能について調べてみよう**
  - **コマンドのマニュアルを表示するには，**man **コマンドを使う（**man**の引数に調べたいコマンド名を指定する）**

```
% man ls ↵
```

カレントディレクトリの移動

- cd（change directoryの略）コマンド
  - 移動したいディレクトリのパス名を引数として指定
  - 引数のディレクトリのパス名は相対パスでも絶対パスでもよい
  - 引数を省略すると，カレントディレクトリをホームディレクトリ（特別教室のMacの場合は，「ホームディレクトリ/CNSiMac」）に変更する

```
% cd Maildir↵
% pwd ↵
/a/fs0102a/t10472ms/Maildir
```

```
% cd /home/t10472ms/Maildir↵
% pwd ↵
/a/fs0102a/t10472ms/Maildir
```

ファイルの内容を見る

- less コマンド
  - テキストファイルの中身を見ることができる
  - 引数に内容を見たいファイル名を指定
  - ファイルをスクロールするには，Spaceキーを使う
  - 閲覧を終了するにはqキーを押す

【演習】
宝探しゲームをしてみよう

- 宝探しゲームをしてみましょう
  - 出発点は
  - cd コマンドでサブディレクトリに移動し，ls コマンドで何があるか調べる
  - ファイルが置いてある場合，宝かどうかmore コマンドでファイルの中身を見る (ハズレの場合もあります)
  - 宝が無いと分かったら，親ディレクトリに移動して別のところを探す cd ..
- 宝を発見したら，宝島の地図 (ディレクトリ構造図) を書いておきましょう

新しいディレクトリを作る

- mkdir（make directoryの略）コマンド
  - 引数に作りたいディレクトリの名前を指定する
  - 正常に作成できると，何も表示されないので，lsコマンドで確認するとよい

```
% mkdir memo ↵
% ls ↵
Desktop         Maildir         XPAppCNS
Wnn             XPDataCNS       test1
test            memo
```

ファイルをコピーする

- cp (copyの略) コマンド
  - コピー元のファイルと新しく作るファイルの名前を空白で区切って引数で指定する
  - 新しく作るファイルの代わりにディレクトリ名を指定すると，そのディレクトリの中に同じ名前で新しいファイルが作成される

```
% ls ↵
fileA    testdir
% cp fileA fileB ↵
% ls ↵
testdir  fileA    fileB
% cp fileA testdir ↵
% ls testdir ↵
fileA
```

ファイルの移動・ファイル名の変更

- mv（moveの略）コマンド
  - ファイルを移動する場合は，mv の後に移動したいファイルの名前，移動先のパス（相対パスか絶対パス）を空白で区切って指定する
  - ファイル名を変更する場合は，mv の後に変更したいファイルの名前，新しいファイル名を空白で区切って指定する

```
% ls ↵
fileA    testdir
% mv fileA testdir ↵
% cd testdir ↵
% ls ↵
fileA
% mv fileA fileB ↵
% ls ↵
fileB
```

※ファイル名の変更の際，変更先のファイル名が既に存在するものであった場合，そのファイルに上書きされ元の内容は消えてしまうので注意

ファイルの削除

- rm（removeの略）コマンド
  - 削除したいファイル名を引数として指定
  - 空白で区切って複数のファイル名を指定できる

```
% ls ↵
fileA fileB
% rm fileA ↵
% ls ↵
fileB
```

ディレクトリの削除

- rmdir (remove directoryの略) コマンド
  - 削除したいディレクトリ名を引数として指定
  - 空白で区切って複数のディレクトリ名を指定できる
  - ディレクトリの下にファイルがある場合，削除できない
  - ディレクトリの下にあるファイルを全て削除するか移動するかした後，ディレクトリを削除する

```
% ls ↵
testdir fileA fileB
% rmdir testdir ↵
% ls ↵
fileA fileB
```

アクセス権と保護モード

アクセス権と保護モード

- CNSの他のユーザのファイル（メールの内容等）を勝手に閲覧されては困るため，適切なアクセス権を設定する必要がある
- ファイルやディレクトリごとに，他のユーザからのアクセスを許可したり，禁止したりする保護モードを設定する機能がある
- 保護モードは「誰が」と「どうする」という組み合わせ（3×3）に対して，許可か禁止かを決めたもの

誰が

1. **ファイルの持ち主のユーザ自身** (user)
2. **グループのメンバ** (group)
   ※ 学生はすべて同じグループに属する
3. **その他** (other)

×

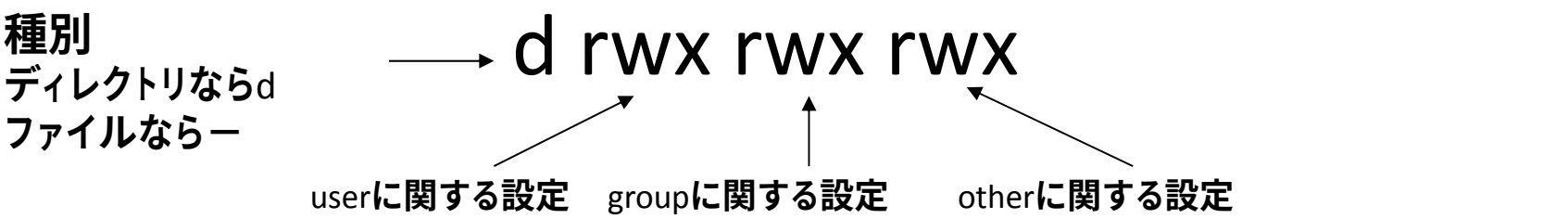
どうする

1. **読み出し** (read)
2. **書き込み** (write)
3. **実行** (execute)

保護モードの確認

- ls -lで保護モードの確認ができる

```
% ls -l↵
-rw-r--r-- 1 t10472ms  student  153  Apr 20 15:30  fileA
```



「r (readの略)」,「w (writeの略)」,「x (executeの略)」それぞれの許可
「ー」は禁止を示す

- 新しく作ったファイルは rw-r--r--になる
- ディレクトリは rwxr-xr-x になる
- メールを保存するディレクトリ (Mail) はrwx------

保護モードの変更

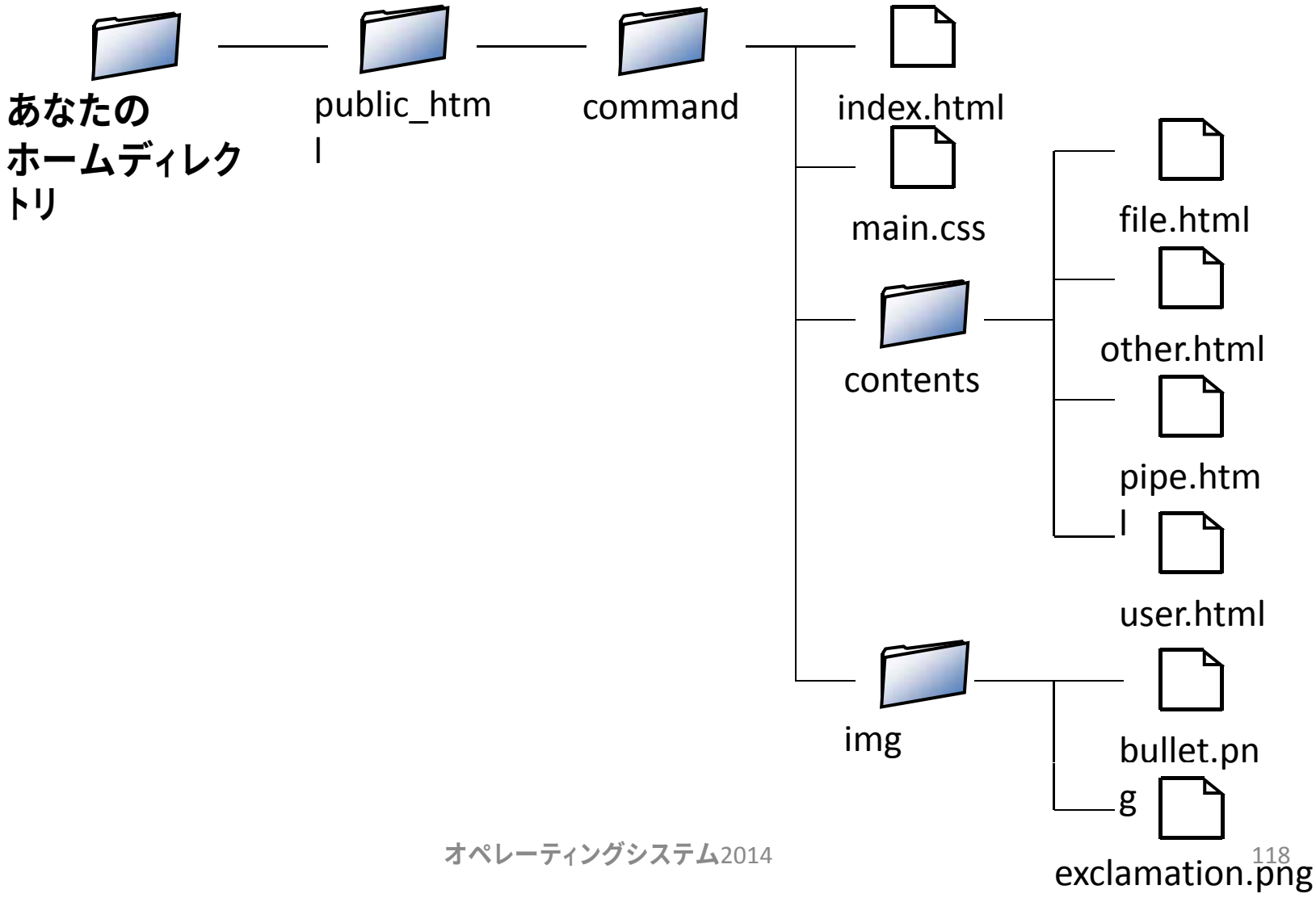
- chmod（change modeの略）コマンド
  - 1番目の引数で，「誰が（u，g，o）」と「どうする」（r，w，x）を＋かーでつないで保護モードを指定（＋は許可，ーは禁止）
  - 2番目の引数で変更したいファイル名またディレクトリ名を指定する

```
% ls -l↵
total 0
-rw-r--r-- 1 t10472ms  student  153  Apr 20 15:30  fileA
% chmod go-r fileA
                  ↵
% ls –l
↵
total 0
-rw------- 1 t10472ms  student  153  Apr 20 15:30  fileA
```

保護モードを設定するときの注意点

- 保護モードを設定するときに注意が必要なディレクトリがある
- 以下のディレクトリは保護モードを変更しない方が安全
  - ホームディレクトリ
    - userに対してexecute権限が必要
    - ログインできなくなる
  - Maildir (メールのデータ)
    - userに対してexecute権限が必要
    - メールが閲覧できなくなる

ディレクトリ構造を作ってみよう



おわり

- **本授業の作成にあたり**
- **慶應大学**SFC IPL/ITB　**岩井クラス**
- **戸辺義人先生**
- **田浦健次朗先生**
- **降旗 大介先生**
- Wikipedia**などの資料を参考にさせていただいています。**

更に学習したい人へ

- http://www.atmarkit.co.jp/fpc/special/raidglossary/raidglossary01.html
- http://pc.nikkeibp.co.jp/article/NPC/20061225/257746/?rt=nocnt
- http://www.raid-119.com/raid.html
- http://www.data-sos.com/raid/raid03.html

ＭＴＴＦ（故障までの平均時間）

- http://www8.plala.or.jp/ap2/shinraisei/shinraisei3.html#mttf

- **有名な計算式**
  - MTTFraid1 = (MTTF x MTTF)/(2 x MTTR)
    - MTTF:**平均故障時間**
    - MTTR:**平均修理時間**

- **ディスクの故障は独立事象か？**
  - **同一機種、同一動作**
- **故障の要因**
  - **（偶発＋環境＋使用）×個体差**
- **故障確率は一定ではない**