UNIX中級講習会

九州大学情報基盤センター

本資料は以下のURLで閲覧できます.
http://www.cc.kyushu-u.ac.jp/scp/lecture/unix2/

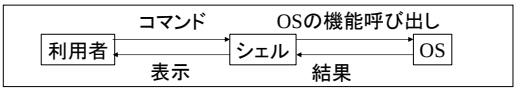
本講習会のスケジュール

- シェルの便利な使い方
  - 高機能シェル Tcsh
  - ファイル名の置換
  - プロセスの管理
- ファイルに関する操作
  - アクセス権の設定
  - ファイルの取りまとめ、圧縮
  - ファイルの探索
  - ファイル内の文字列探索
- 一括処理
  - シェルスクリプト
  - Make
- Emacsのバッファ操作

シェルの機能

- シェル ＝ 利用者の応対係
  - 基本的な役割:
    - コマンド受付のプロンプトを表示
    - キーボードからコマンド受付
    - コマンドに対応する OS 機能の呼び出し
    - OSからの結果を画面に表示
  - 便利な機能
    - ファイル名の一部だけでファイルを指定する
    - 起動されたプロセスを"ジョブ"として管理する
    - 条件分岐や繰り返しを使って複雑な処理を実現する

  etc.



Tcsh: 高機能シェル

- 通常のシェルに様々な機能を追加し,
  より便利になったシェル.
  - 補完機能
    コマンドやファイル名を途中まで入力し,
    TABキーを押すと残りを補完する.
  - コマンド履歴の呼び出し機能
    今まで入力したコマンドを矢印キーの上下で
    呼び出す.
  - コマンド行編集機能
    入力中のコマンドの書き換え、
    カットアンドペースト等.

Tcshの利用法

- 開始: tcsh コマンドで起動
- 終了: exit (元のシェル csh に戻る)
- 通常のコマンドに加え,以下の機能を利用可能

キー操作	機能
→ (もしくは C-f)	カーソルを一つ右に移動
← (もしくは C-b)	カーソルを一つ左に移動
C-a	左端に移動
C-e	右端に移動
C-u	全部削除
C-k	カーソルから右側を切り取り
C-y	直前に切り取った部分を貼り付け
↑ (もしくは C-p)	実行したコマンド履歴の古い方へ
↓ (もしくは C-n)	実行したコマンド履歴の新しい方へ
TABキー	コマンドやファイルの補完

ファイル名の置換 ＊

- 名前の一部だけでファイルやディレクトリを指定
  - 例1) ファイル名の最初が test であるファイルを消去

```
kyu-cc% rm test*
```

  - 例2) ファイル名の末尾が .f90 である
    ファイルを fortran ディレクトリに移動

```
kyu-cc% mv *.f90 fortran
```

  - **例3) ファイル名の最初が hello で末尾が .c のファイルを helloディレクトリにコピー**

```
kyu-cc% cp hello*.c hello
```

  - **例4) カレントディレクトリの下の全てのディレクトリの下のファイルのうち名前の末尾が .f90 のものを test ディレクトリにコピー**

```
kyu-cc% cp */*.f90 test
```

注意： 誤って消去したり 上書きしたりしないように

- 実行前に ls コマンドで他のファイル名も確認.
- cp, mv, rm コマンドには -i オプションを付け，消去，上書きされるファイル名を確認.

実習1

実習1： Tcshの利用及び ファイル名の置換

- Tcshの機能の確認
- ＊を利用したファイル操作

ジョブの管理

- 注意： バッチ処理のジョブとは違う.
  - シェルのジョブは
    - 全ての UNIXで使用可能.
    - 基本的に，対話型利用.
    - 課金も対話型（於 本センター）.
    - ログアウトすると強制終了される. （強制終了されずに残ってしまう場合もある）
  - バッチ処理のジョブは
    - 主にセンターのような共同利用施設の計算機で使用.
    - 基本的にバッチ型利用.
    - 課金もバッチ型（於 本センター）.
    - ログアウトしても終了しない.

シェルのジョブの状態

- フォアグラウンド
  - 直接端末から入出力を行える状態.
- バックグラウンド
  - フォアグラウンドジョブの裏で同時に実行している状態
- 中断
  - 実行中に一時停止された状態
- 終了
  - 終了した、または終了させられた状態

ジョブ管理

- バックグラウンド
  - 見かけ上、同時に複数の処理を実行

通常:

```
kyu-cc%  ./a.out
```

バックグラウンド: コマンドの後ろに & を付ける

```
kyu-cc%  ./a.out  &
kyu-cc%
```

注意:
ログアウトしてもバックグラウンドジョブが残ることがあるので、全てのバックグラウンドジョブを終了させてからログアウトする

ジョブ管理のコマンド、キー操作

- コマンド

`fg`	中断状態のジョブを再開
`bg`	中断状態のジョブをバックグラウンドで再開
`kill -KILL %番号`	指定した番号のジョブを強制終了
`jobs`	バックグラウンドで実行中のジョブ，及び中断状態のジョブを表示

- キー操作

`C-z`	フォアグラウンドジョブを中断
`C-c`	フォアグラウンドジョブを強制終了

ジョブ管理：使用例

- 一旦起動したプログラムをバックグラウンドに変更

```
kyu-cc%  ./hello
  C-z ←Ctrlキーを押しながらz
[1]  + 19277 Suspended              ./hello
kyu-cc%  bg
[1]    ./hello &
kyu-cc%
```

- 不要なジョブを強制終了

```
kyu-cc%  jobs
[1]    Running                      ./hello
[2]  + Running                      ./a.out
[3]  - Running                      emacs
kyu-cc%  kill  -KILL  %2
kyu-cc%
```

注意：

- キーボードからの入力や画面への出力を行うプログラムはそのままではバックグラウンドで実行できない．
  - プログラムへのデータ入力とコマンド入力，コマンドプロンプト kyu-cc% とプログラムの出力が混ざって使いづらいため．
  - バックグラウンドで実行しようとすると，強制的に中断状態になる．
  - 以下のようにリダイレクションを使って対処．

```
kyu-cc%  ./a.out < data > log &
```

実習2

実習2： ジョブ管理

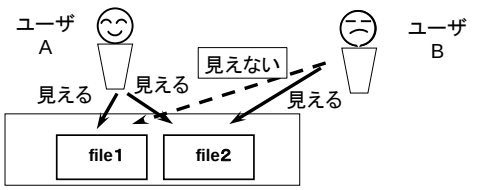
- ジョブの起動、中断、再開、終了

本講習会のスケジュール

- シェルの便利な使い方
  - 高機能シェル Tcsh
  - ファイル名の置換
  - プロセスの管理
- ファイルに関する操作
  - アクセス権の設定
  - ファイルの取りまとめ、圧縮
  - ファイルの探索
  - ファイル内の文字列探索
- 一括処理
  - シェルスクリプト
  - Make
- Emacsのバッファ操作

ファイルのアクセス権

- ファイルごとにアクセス権を指定可能



ファイルのアクセス権

- 各ファイルについて
  - **ユーザのカテゴリ**毎に
  - **アクセス権**を

  設定可能

ユーザのカテゴリ

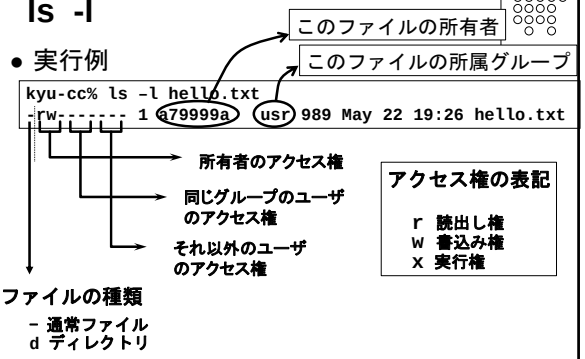
- •ファイルの所有者
- •同じグループ
- •それ以外

アクセス権

- •読み出し権
- •書き込み権
- •実行権

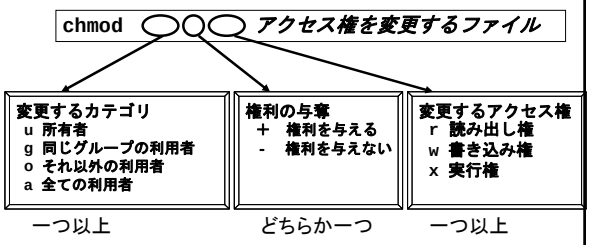
ファイルのアクセス権表示
ls -l

- 実行例



アクセス権の変更
chmod

- 利用方法



アクセス権の変更
chmod

- 利用例
  - 例1）ファイルの所有者に実行権を与える

```
kyu-cc%  chmod u+x hello.txt
```

  - 例2）同じグループのユーザーに読み出し権と書き込み権を与える

```
kyu-cc%  chmod g+rw hello.txt
```

  - 例3）所有者以外の全てのアクセスを禁止

```
kyu-cc%  chmod go-rwx hello.txt
```

ディレクトリのアクセス権

- ファイルのアクセス権とディレクトリのアクセス権は少し意味が異なる.
  - 読み出し権:
    そのディレクトリにあるファイルの名前を閲覧する権利
  - 書き込み権:
    そのディレクトリでファイルの削除や新規作成をする権利
  - 実行権:
    そのディレクトリに移動する権利.
    ただし, 実行権が無ければ閲覧もファイルの編集・削除・新規作成も出来ない.

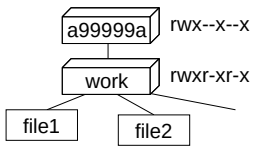
実習3

実習3: アクセス権

- ファイルとディレクトリのアクセス権の操作
  - 読み出し権
  - 書き込み権
  - 実行権

アクセス権設定の応用:
限られた利用者だけでファイルを共有

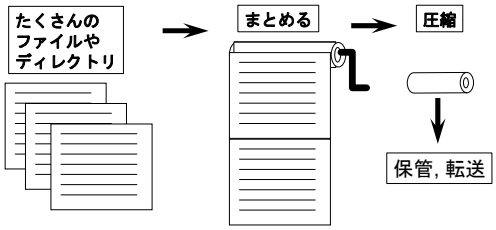
- グループはシステムの管理者でなければ設定できないので, センターでは使えない.
  → アクセス権を利用し, アクセスできる利用者を限定する.



•work というディレクトリ名は所有者にしか見えない
•しかし, すべてのユーザーが work に移動可能
•すなわち, work のパスを知っているユーザーだけが file1, file2 を参照できる

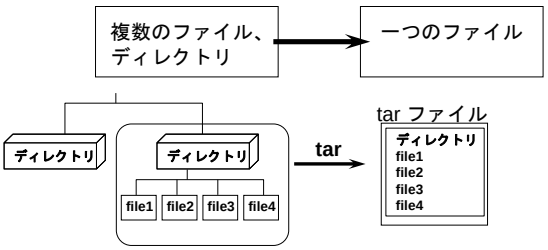
ファイルのとりまとめと圧縮

- 多数のファイルを保管したり他の計算機に転送するとき，
  一つのファイルにまとめて，圧縮（サイズを小さく）
  すると便利．



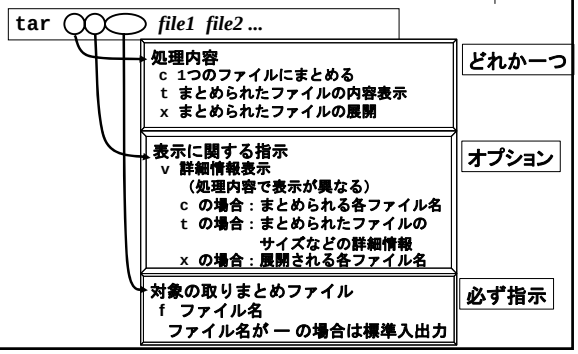
ファイルの取りまとめと展開 tar

- 複数のファイルを一つにまとめる
  - 本来はテープに保存する（tape archive）コマンド



ファイルの取りまとめと展開 tar

- 利用法



ファイルのとりまとめと展開 tar

- **利用例**
  - **例1）カレントディレクトリの work と tmp ディレクトリを test.tar というファイルにまとめる．**

```
kyu-cc%  tar cf test.tar work tmp
```

ファイル名の末尾を .tar としておくと tar でまとめたファイルであることが分かって便利です．

  - **例2）test.tar の中身を表示（詳細情報も表示）**

```
kyu-cc%  tar tvf test.tar
```

  - **例3）test.tar を元の形に展開（展開される各ファイルも表示）**

```
kyu-cc%  tar xvf test.tar
```

tar でディレクトリをまとめるときの注意

- ディレクトリは相対パスで指定した方が良い

```
kyu-cc%  tar cvf test.tar test
```

  - 任意の位置に展開することができる
    → バックアップや他の計算機での利用が容易

  - もし絶対パスで指定した場合

```
kyu-cc%  tar cvf test.tar /home/user3/k70043a/test
```

全く同じ絶対パスにしか展開できない。

tar ファイルを展開するときの注意

- 展開時に同じパスのファイルがあると上書きする．

- 対策1：
  - 事前に

```
tar tvf tarファイル
```

で tarファイルの内容を確認したり

```
ls -R
```

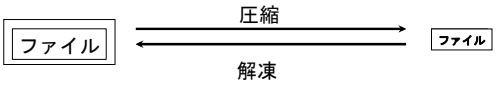
で現在カレントディレクトリ以下にあるファイルを調べる．

- 対策2：
  - 一旦，新しいディレクトリを作成してその中で展開し，必要なものをコピーする．

```
mkdir tmp
cd tmp
tar xvf tarファイル
```

ファイルの圧縮, 解凍

- 圧縮：ファイル中の冗長な部分を削って
  サイズを小さくする.
- 解凍：削られた冗長部分を補って
  元のファイルに戻す.



ファイルの圧縮, 解凍
gzip

- 利用法： 圧縮

  gzip　圧縮するファイル名
  - 圧縮結果は, 元のファイル名の末尾に .gz が付いたファイルに格納される.
  - 例） hello.txt を圧縮する.（hello.txt.gz が作成される.）

  ```
  kyu-cc%  gzip hello.txt
  ```
- 利用法： 解凍

  gzip -d　解凍するファイル名
  - 例） hello.txt.gz を解凍する.（hello.txt が作成される.）

  ```
  kyu-cc%  gzip -d hello.txt.gz
  ```

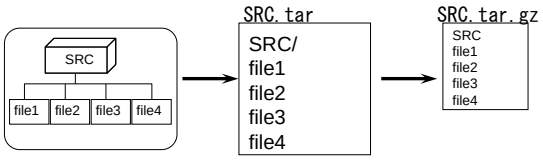
注意: 圧縮,解凍コマンド実行後, 元のファイルは消える

圧縮するファイルの選択

- ファイルを圧縮する利点
  - ファイルの転送が速い
  - ディスクを有効に利用できる
- 不便な点
  - 通常､使用する前に復元（解凍）が必要

通常, 他の計算機に転送するファイルや
しばらく利用しないファイルを圧縮する

tar と gzip を組み合わせると効果的

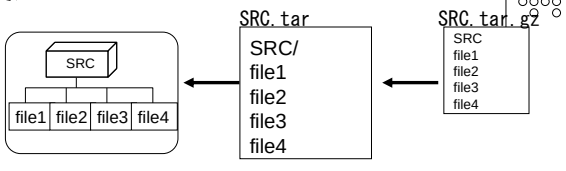


tar cvf SRC.tar SRC
SRC.tar ファイル
ができる

gzip SRC.tar
SRC.tar.gzファイル
ができる

tar cvf - SRC | gzip > SRC.tar.gz
一度に実行

取りまとめて圧縮されたファイルの復元



tar xvf SRC.tar
SRCディレクトリ以下が
展開される

gzip -d SRC.tar.gz
SRC.tarファイル
ができる

gzip –dc SRC.tar.gz | tar xvf -
一度に実行

gzip –dc ファイル名
解凍結果を標準出力に出力

ファイルの探索

- ファイルやディレクトリが増えてくると,
  目的のファイルがどこにあるかわからなくなってくる.
  → 探索コマンド find

ファイルの探索 find

- 利用法

```
find 探索開始ディレクトリ –name 名前 探索後の処理
```

  - 探索開始ディレクトリは絶対パス,相対パスどちらで指定してもよい.
  - 主な”探索後の処理”:
    -print　見つかったファイルを表示
    -exec　見つかったファイルに対して コマンドを実行

ファイルの探索 find

- 利用例
  - 例1） work ディレクトリ以下から proj1.f90 を 探して表示

```
kyu-cc% find work –name proj1.f90 -print
```

  - 例2） ホームディレクトリ以下から 末尾が~であるファイルを探して削除

```
kyu-cc% find ~ -name '*~' –exec rm –i {} ¥;
```

    - コマンドには,見つかったファイル名を {} で指定する.
    - コマンドの末尾は”空白を一つ以上置いて” ¥;

文字列の探索

- ファイル内から特定の文字列を含む行を探して表示する.
- プログラムのデバッグなどに利用

文字列の探索 grep

- 利用法

```
grep オプション 検索文字列 ファイル
```

  - 主なオプション
    –i　大文字小文字を区別しない
    –n　行番号も表示する
    –l　指定した文字列を含む ファイルの名前だけを表示する （複数のファイルを指定した場合に利用）
    –v　指定した文字列を含まない行を表示する.

文字列の探索 grep

- 利用例
  - 例1） ファイル test.c から total という文字列 を含む行を探索する.

```
kyu-cc%  grep –i total test.c
```

  - 例2） ファイル名の末尾が .f90 のファイルから, ! を含まない行を探索する.

```
kyu-cc%  grep –v '¥!' *.f90
```

    - ! のような特殊記号は '¥!' のように ¥を付け, ' で囲んで指定する.

実習4

演習4: ファイル操作

- ファイルの取りまとめ、展開
- ファイルの圧縮、解凍
- ファイルの検索
- ファイル内の文字列検索

本講習会のスケジュール

- シェルの便利な使い方
  - 高機能シェル Tcsh
  - ファイル名の置換
  - プロセスの管理
- ファイルに関する操作
  - アクセス権の設定
  - ファイルの取りまとめ、圧縮
  - ファイルの探索
  - ファイル内の文字列探索
- 一括処理
  - シェルスクリプト
  - Make
- Emacsのバッファ操作

シェルスクリプト

- 一連のコマンドをファイルに列挙し一括して実行
  - 条件分岐,繰り返しなども利用可能
- 用途:
  - 何度も行う決まった処理
  - 複雑な処理
  - バッチジョブとして投入する処理

シェルスクリプト

- 利用法

```
スクリプトファイルのパス オプション
```

  - カレントディレクトリにある場合は ./ を忘れない.
  - 引数を受け付けるかどうかは, シェルスクリプトの内容による.
  - コマンドとして実行する前に chmod で実行権限を与えておく.

```
kyu-cc%  chmod u+x スクリプトファイル名
```

シェルスクリプトの例1

- workディレクトリに移動し, test.f90 をコンパイルして実行する.
  - バッチジョブのスクリプト

```
#!/bin/csh
cd  work
frt  test.f90  -o test
./test
```

シェルスクリプトの例2

- 引数に指定されたファイルが存在すれば, そのファイルで member という文字列を探索する.

```
#!/bin/csh
if  (-f  $1)  then
  grep  'member'  $1
else
  echo  "$1 doesn't exist."
endif
```

  - 引数は $数字 で指示する. $1 : 1番目の引数.
  - ファイルが存在するか否かの判定 **if (-f ファイル名)**
  - **メッセージの出力 echo メッセージ**

シェルスクリプトの例3

- カレントディレクトリにある, ファイル名の末尾が .c であるファイルに対し, ファイル名の末尾にさらに .old を付ける.

```
#!/bin/csh
foreach  cfile  ( *.c )
  mv  ${cfile}  ${cfile}.old
  echo  ${cfile}  ${cfile}.old
end
```

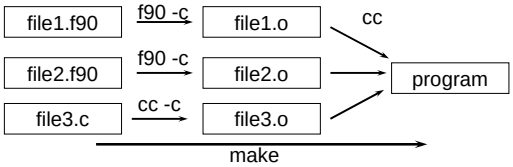
  - 繰り返しは foreach 変数 (変数のとる値) から end まで.
  - ここでは変数のとる値として *.c 、すなわち カレントディレクトリにある、ファイル名の末尾が .c のファイルの名前を指定している.
  - 条件を満たすファイル名について,順に繰り返しの中を実行する.
  - 変数の中身は $[変数]で参照.

分割コンパイルの支援

- 分割コンパイル
  - 一つのプログラムを複数のファイルに分割
    → 大規模なプログラムの管理が容易
  - 一部のファイルに修正を施した後,
    再度全部をコンパイルするのは無駄.
    → 修正を行ったファイルのみコンパイル

分割コンパイルの支援 make

- ファイルの依存関係を元に
  必要なコンパイルだけを行う仕組み.



- ファイルの依存関係は Makefile というファイルに記述.

Makefileの例

- FortranとC言語の混成プログラム.
- コンパイル以外のターゲットも作成可能
  例）
  - clean: 途中でできた不要ファイルを削除
  - backup: 必要なファイルをまとめて圧縮

```
all: program
program: file1.o file2.o file3.o
        cc -o program file1.o file2.o file3.o
file1.o: file1.f90
        f90 -c file1.f90
file2.o: file2.f90
        f90 -c file2.f90
file3.o: file3.c
        cc file3.c
clean:
        rm -f file1.o file2.o file3.o *~
backup: files.tar.gz
files.tar.gz: file1.f90 file2.f90 file3.c
        tar -cf - file1.f90 file2.f90 file3.c | gzip > files.tar.gz
```

Makefileの書き方

- 基本形

```
ターゲット:　ターゲット作成に必要なファイル
TAB　アクション
```

  - ターゲット: 目的
    - ファイル,もしくは単なる名前
  - ターゲット作成に必要なファイル名:
    - 複数指定可
  - アクション:
    - ターゲットを作成するスクリプト
    - 複数行にまたがっても可
    - 行の先頭は必ず TAB

分割コンパイルの支援 make

- 利用法

```
make ターゲット
```

  - ターゲット: 何を生成するか
    →Makefileに記述されているものを指定する.
  - 何もターゲットが指定されない場合,
    all が指定されたものとして実行される.

実習5

演習5: 一括処理

- シェルスクリプトによる一括処理
  - サンプルの実行
  - （応用） 簡単なシェルスクリプトの作成
- Makeの利用
  - Makefile の例
  - 一括コンパイル

本講習会のスケジュール

- シェルの便利な使い方
  - 高機能シェル Tcsh
  - ファイル名の置換
  - プロセスの管理
- ファイルに関する操作
  - アクセス権の設定
  - ファイルの取りまとめ、圧縮
  - ファイルの探索
  - ファイル内の文字列探索
- 一括処理
  - シェルスクリプト
  - Make
- Emacsのバッファ操作

Emacsのバッファ操作

- バッファ:
  Emacs内で扱われるファイルのイメージ
- Emacsでは複数のバッファの扱いが可能.
- さらに，画面を分割して
  複数のバッファの内容を同時に
  表示することが可能.

Emacsの操作

- Ctrlキーを押しながら… の操作
  - C-x … Ctrl キーを押しながら x を押す.
  - C-BackSpace ... Ctrl キーを押しながら
    BackSpaceキーを押す

バッファの作成
C-x C-f ファイル名

- 指定した名前のバッファを作成
- 指定した名前のファイルが存在すれば
  そのファイルの内容を読み込んで編集.
  - 存在しなければ空の状態から編集開始.
- 同じ名前のバッファがすでに存在する場合:
  - 対応するファイルも同じならそのバッファに切り替え
  - 対応するファイルが異なるなら、バッファ名は
    ファイル名 <2>
    のようにファイル名に作成順の番号が付く.

バッファの切り替え
C-x b バッファ名

- 指定したバッファに切り替える.
- バッファ名を指定する前に TABキーを押すと
  選択可能なバッファ名の一覧を表示

バッファの削除
C-x k バッファ名

- バッファ名を省略して改行すると,
  直前に扱っていたバッファが削除される.
- バッファの内容を書き換えた後保存していなければ,
  ファイルに保存するか否かを尋ねられるので
  保存したい場合は y を,
  保存したくない場合は n を入力.
- 上記で n を入力すると,本当に削除してよいか
  尋ねられるので yes と入力.

画面の分割

- 現在カーソルのある画面を分割
  - C-x 2
    画面を上下に分割
  - C-x 3
    画面を左右に分割
- 画面を移動
  - C-x o
    カーソルを次の画面に移動
- 画面を減らす
  - C-x 0
    現在カーソルのある画面を削除
    （バッファは残る）
  - C-x 1
    現在カーソルのある画面を残し,他の画面を削除

Emacsのコマンド

キー操作	意味
C-x C-f	新しくバッファを作成
C-x b	バッファの切り替え
C-x 2	画面を上下に分割
C-x 3	画面を左右に分割
C-x 1	現在カーソルのある画面のみにする
C-x 0	現在カーソルのある画面を消す
C-x o	次の画面に移る
C-z	Emacs の中断

実習6

演習6: Emacsのバッファ操作

- バッファの作成、切り替え

お疲れ様でした

- 実習で利用した ID は 1週間利用できます.
- 不明な点等は
  request@cc.kyushu-u.ac.jp
  もしくは
  nanri@cc.kyushu-u.ac.jp
  まで.