

みんなのPython勉強会 #19

2016.12.7

シェルでコンピュータ を操ろう！

辻 真吾

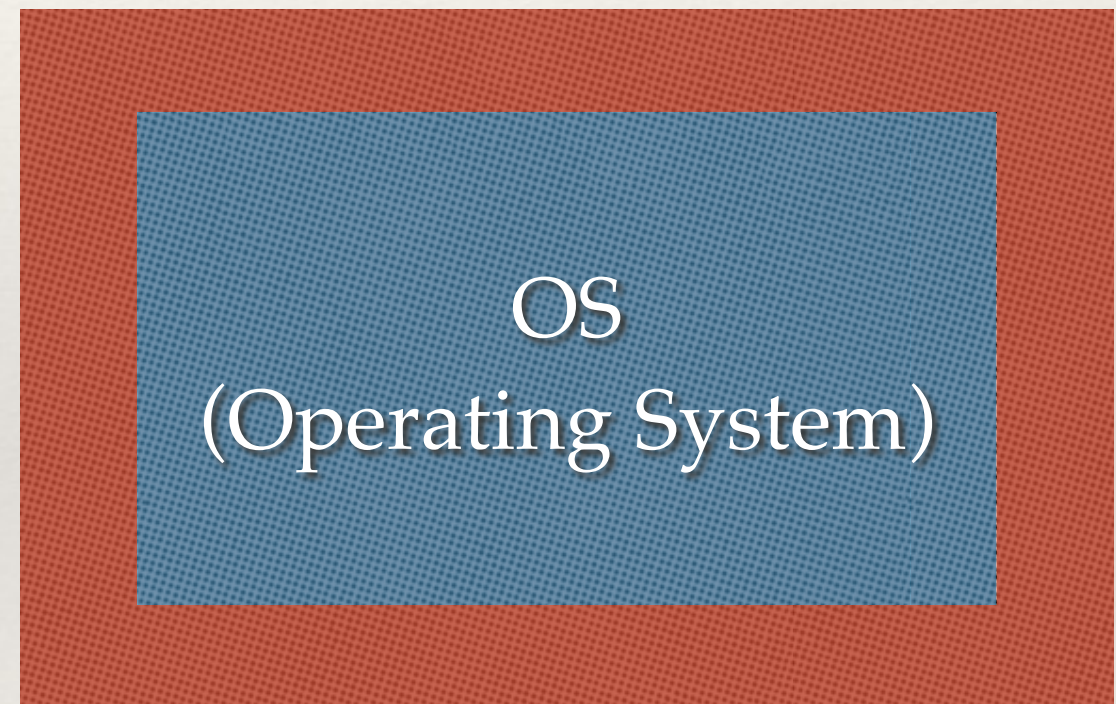
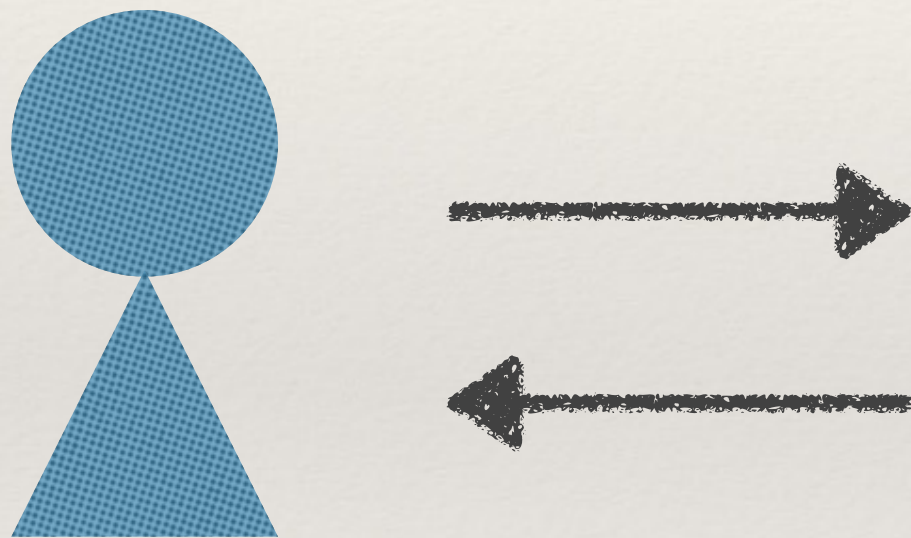
@tsjshg

自己紹介

- ❖ 1975年生まれ
- ❖ 都内のある大学で研究やってることになってます
 - ❖ 研究室のテーマは癌とゲノム
 - ❖ 私がやっているのはPythonで主に機械学習を使ったデータ解析
- ❖ Udemyで「実践Pythonデータサイエンス」やってます
- ❖ AI活用時代にPythonで見る夢@CTC教育サービス
 - ❖ <http://www.school.ctc-g.co.jp/columns/tsuji/>
- ❖ <http://www.tsjshg.info/>

今日はシェルの話

シェルとは？



シェル (shell)

bash

- ❖ Bourne Again Shell
- ❖ 70年代に普及したBourne Shellの機能強化版
- ❖ macOSや多くのLinuxでは、ターミナル
- ❖ Windows10から、Microsoft社がbashを供給
- ❖ 開発者モードが必要



```
File: bash.info, Node: Top, Next: Introduction, Prev: (dir), Up: (dir)

Bash Features
*****

This text is a brief description of the features that are present in
the Bash shell (version 3.2, 28 September 2006).

This is Edition 3.2, last updated 28 September 2006, of 'The GNU
Bash Reference Manual', for 'Bash', Version 3.2.

Bash contains features that appear in other popular shells, and some
features that only appear in Bash. Some of the shells that Bash has
borrowed concepts from are the Bourne Shell ('sh'), the Korn Shell
('ksh'), and the C-shell ('csh' and its successor, 'tcsh'). The
following menu breaks the features up into categories based upon which
one of these other shells inspired the feature.

This manual is meant as a brief introduction to features found in
Bash. The Bash manual page should be used as the definitive reference
on shell behavior.

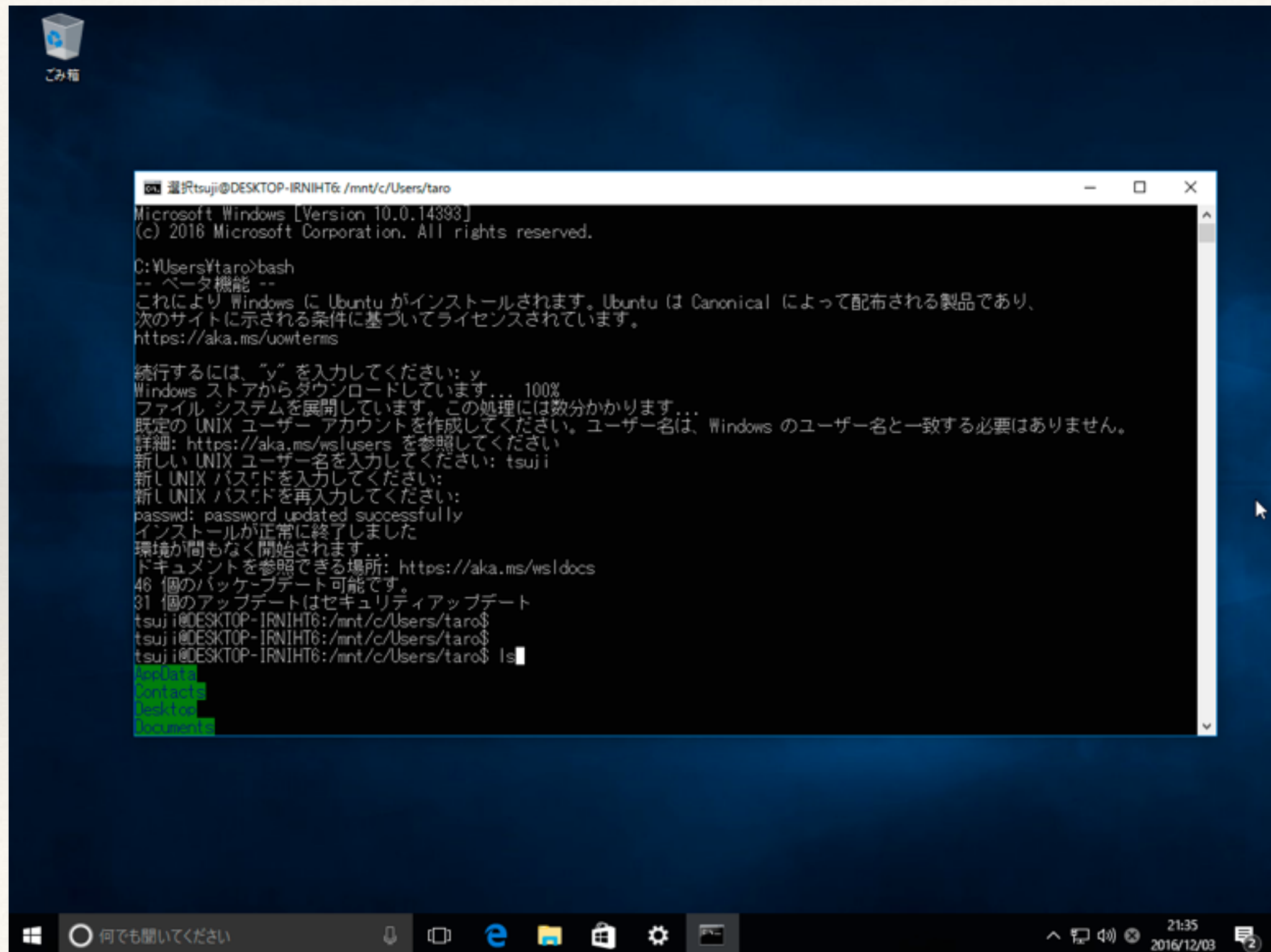
* Menu:

* Introduction::          An introduction to the shell.
* Definitions::          Some definitions used in the rest of this
                          manual.
* Basic Shell Features::  The shell "building blocks".
* Shell Builtin Commands:: Commands that are a part of the shell.
* Shell Variables::      Variables used or set by Bash.
* Bash Features::        Features found only in Bash.
* Job Control::          What job control is and how Bash allows you
                          to use it.
* Using History Interactively:: Command History Expansion
* Command Line Editing::  Chapter describing the command line
                          editing features.
* Installing Bash::       How to build and install Bash on your system.
* Reporting Bugs::        How to report bugs in Bash.
* Major Differences From The Bourne Shell:: A terse list of the differences
                                          between Bash and historical
                                          versions of /bin/sh.

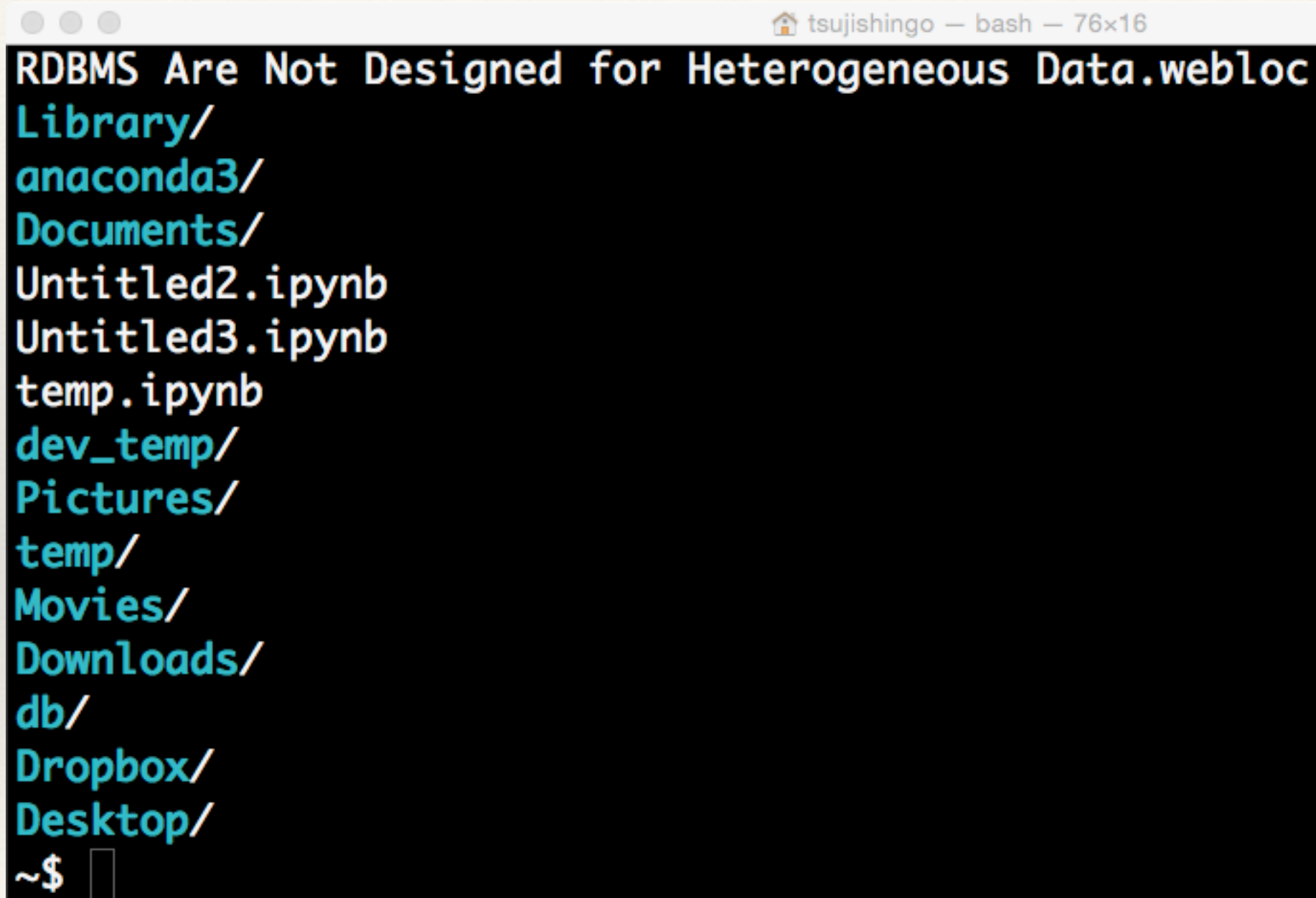
* Copying This Manual::   Copying this manual.
* Builtin Index::         Index of Bash builtin commands.
* Reserved Word Index::   Index of Bash reserved words.
* Variable Index::        Quick reference helps you find the
                          variable you want.
* Function Index::        Index of bindable Readline functions.
* Concept Index::         General index for concepts described in
                          this manual.

-----Info: (bash.info)Top, 50 行 --All----*** 旧式のタグ ***-----
Info バージョン 4.8 によるこそ。? で使い方、m でメニュー項目を呼び出せ?
```

Bash on Ubuntu on Windows



ls



A terminal window titled "tsujishingo — bash — 76x16" displays the output of the "ls" command. The output lists various files and directories in a color-coded format: directories are in cyan, files are in white, and executables are in yellow. The list includes "RDBMS Are Not Designed for Heterogeneous Data.webloc", "Library/", "anaconda3/", "Documents/", "Untitled2.ipynb", "Untitled3.ipynb", "temp.ipynb", "dev_temp/", "Pictures/", "temp/", "Movies/", "Downloads/", "db/", "Dropbox/", and "Desktop/". The prompt "~\$ " is visible at the bottom left.

```
tsujishingo — bash — 76x16
RDBMS Are Not Designed for Heterogeneous Data.webloc
Library/
anaconda3/
Documents/
Untitled2.ipynb
Untitled3.ipynb
temp.ipynb
dev_temp/
Pictures/
temp/
Movies/
Downloads/
db/
Dropbox/
Desktop/
~$
```

まずは、超基本から

ls -lrt

```
tsujishingo — bash — 76x16
d for Heterogeneous Data.webloc
drwx-----@ 60 tsujishingo staff 2040 10 10 12:33 Library/
drwxr-xr-x 25 tsujishingo staff 850 10 10 15:36 anaconda3/
drwx-----+ 29 tsujishingo staff 986 11 2 08:41 Documents/
-rw-r--r-- 1 tsujishingo staff 948 11 13 16:54 Untitled2.ipynb
-rw-r--r-- 1 tsujishingo staff 622 11 13 16:55 Untitled3.ipynb
-rw-r--r-- 1 tsujishingo staff 120265 11 20 01:00 temp.ipynb
drwxr-xr-x 4 tsujishingo staff 136 11 23 23:43 dev_temp/
drwx-----+ 5 tsujishingo staff 170 11 27 10:19 Pictures/
drwxr-xr-x 27 tsujishingo staff 918 11 27 16:18 temp/
drwx-----+ 35 tsujishingo staff 1190 12 1 00:01 Movies/
drwx-----+ 704 tsujishingo staff 23936 12 3 19:29 Downloads/
drwx-----@ 28 tsujishingo staff 952 12 4 10:29 db/
drwx-----@ 83 tsujishingo staff 2822 12 4 10:30 Dropbox/
drwx-----+ 4 tsujishingo staff 136 12 4 10:31 Desktop/
~$
```

オプション3つ。lで詳しく、tで更新時間順、それをrでひっくり返す

11

```
tsujishingo — bash — 76x16
drwx-----+  5 tsujishingo  staff  170B  11  27  10:19 Pictures/
drwxr-xr-x   27 tsujishingo  staff  918B  11  27  16:18 temp/
drwx----- 251 tsujishingo  staff  8.3K  11  29  09:11 .Trash/
drwx-----+ 35 tsujishingo  staff  1.2K  12   1  00:01 Movies/
drwxr-xr-x   4 tsujishingo  staff  136B  12   2  09:36 .matplotlib/
drwx-----+ 704 tsujishingo  staff   23K  12   3  19:29 Downloads/
-rw-----   1 tsujishingo  staff  5.1K  12   4  00:32 .viminfo
-rw-----   1 tsujishingo  staff  8.2K  12   4  02:08 .bash_history
drwx-----@ 28 tsujishingo  staff  952B  12   4  10:29 db/
drwx-----@ 83 tsujishingo  staff  2.8K  12   4  10:30 Dropbox/
drwx-----+ 4 tsujishingo  staff  136B  12   4  10:31 Desktop/
drwxr-xr-x+  88 tsujishingo  staff  2.9K  12   4  12:28 ./
-rw-r--r--@  1 tsujishingo  staff   38K  12   4  12:28 .DS_Store
~$ alias ll
alias ll='ls -rtlhaGv'
~$
```

ll という別名 (alias) で呼べる。alias コマンドの引数で詳細を確認できる。

`alias ll='ls -rtlhaGv'` ← alias を作りたいときはこんな感じ。

man ls

- ❖ manコマンドの引数に、調べたいコマンド名を書く
- ❖ 画面が閲覧モードに切り替わるので、Qで復帰
- ❖ 今はググった方がいいか？

```
19th_ -- less -- 100x58
LS(1) BSD General Commands Manual LS(1)
NAME
  ls -- list directory contents
SYNOPSIS
  ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]
DESCRIPTION
  For each operand that names a file of a type other than directory, ls displays its
  name as well as any requested, associated information. For each operand that names a
  file of type directory, ls displays the names of files contained within that direc-
  tory, as well as any requested, associated information.

  If no operands are given, the contents of the current directory are displayed. If
  more than one operand is given, non-directory operands are displayed first; directory
  and non-directory operands are sorted separately and in lexicographical order.

  The following options are available:

  -l      Display extended attribute keys and sizes in long (-l) output.

  -1      (The numeric digit ``one''.) Force output to be one entry per line. This is
  the default when output is not to a terminal.

  -A      List all entries except for . and ... Always set for the super-user.

  -a      Include directory entries whose names begin with a dot (.).

  -B      Force printing of non-printable characters (as defined by ctype(3) and cur-
  rent locale settings) in file names as \xxx, where xxx is the numeric value
  of the character in octal.

  -b      As -B, but use C escape codes whenever possible.

  -C      Force multi-column output; this is the default when output is to a terminal.

  -c      Use time when file status was last changed for sorting (-t) or long printing
  (-l).

  -d      Directories are listed as plain files (not searched recursively).

  -e      Print the Access Control List (ACL) associated with the file, if present, in
  long (-l) output.

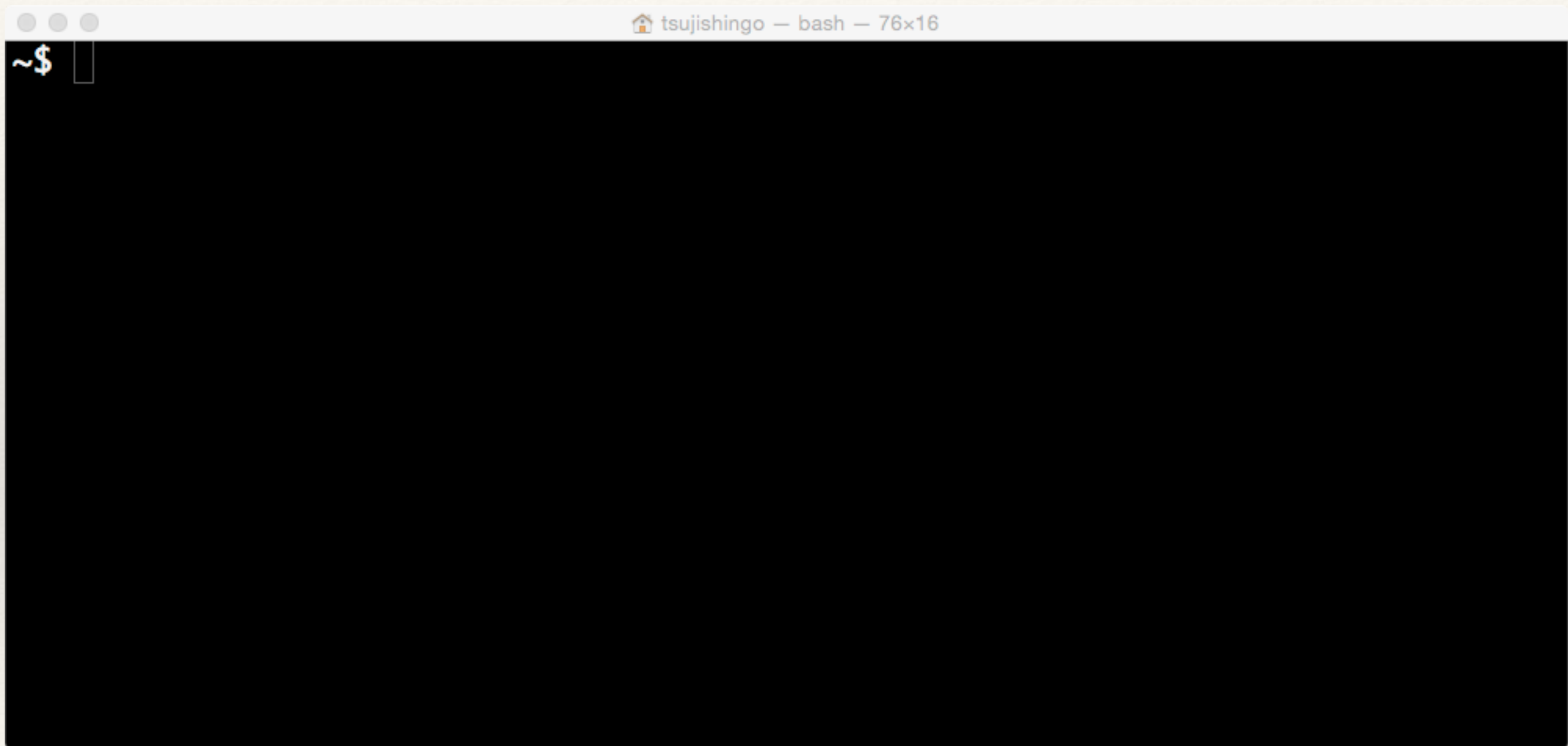
  -F      Display a slash (/) immediately after each pathname that is a directory, an
  asterisk (*) after each that is executable, an at sign (@) after each
  symbolic link, an equals sign (=) after each socket, a percent sign (%)
  after each whiteout, and a vertical bar (|) after each that is a FIFO.

  -f      Output is not sorted. This option turns on the -a option.

  -G      Enable colorized output. This option is equivalent to defining CLICOLOR in
  the environment. (See below.)

  -g      This option is only available for compatibility with POSIX; it is used to
```

clear



画面を綺麗にしてくれる。Ctrl+Lでも同じ事ができます。

移動の基本

```
tsujishingo — bash — 76x16
~$
~$ pwd
/Users/tsujishingo
~$ mkdir demo
~$ cd demo
~/demo$ pwd
/Users/tsujishingo/demo
~/demo$ cd ..
~$ cd de
demo/      dev_temp/
~$ cd demo
~/demo$ cd
~$ pwd
/Users/tsujishingo
~$
```

今居る場所

demoディレクトリを作って、cdで移動

..は1つ上

tabで補完。候補一覧の表示

引数なしでホームに戻る

ファイルのコピー

```
demo — bash — 76x16
~$ ls *.pdf
comp.pdf  hoge.pdf  temp.pdf  test.pdf
~$ cp *.pdf de
demo/      dev_temp/
~$ cp *.pdf demo/
~$ cd demo
~/demo$ ll
total 320
drwxr-xr-x+ 89 tsujishingo  staff  3.0K 12  4 12:48 ../
-rw-r--r--@  1 tsujishingo  staff   68K 12  4 12:56 test.pdf
-rw-r--r--@  1 tsujishingo  staff   9.3K 12  4 12:56 temp.pdf
-rw-r--r--@  1 tsujishingo  staff   45K 12  4 12:56 hoge.pdf
-rw-r--r--@  1 tsujishingo  staff   31K 12  4 12:56 comp.pdf
drwxr-xr-x  6 tsujishingo  staff  204B 12  4 12:56 ./
~/demo$ ls
comp.pdf  hoge.pdf  temp.pdf  test.pdf
```

ホームディレクトリにあるpdf

これを、demoディレクトリにコピー

cpのところ、mvだと移動になります。

削除

```
tsujishingo — bash — 76x16
~/demo$
~/demo$ rm hoge.pdf
remove hoge.pdf? y
~/demo$ rm -f te
temp.pdf test.pdf
~/demo$ rm -f test.pdf
~/demo$ cd
~$ rm -rf demo
~$
```

ファイルを1つ消す

確認しない

確認しないで、ディレクトリまるごと消す

history

```
tsujishingo — bash — 76x16
508  ls *.png
509  ls *.py
510  ls *.pdf
511  ls *.pdf
512  cp *.pdf demo/
513  cd demo
514  ll
515  ls
516  rm hoge.pdf
517  rm -f test.pdf
518  cd
519  rm -rf demo
520  history
~$ !518
cd
~$
```

↑↓で、このコマンド履歴を辿っていきます。

Ctrl+R

```
tsujishingo — bash — 76x16
508  ls *.png
509  ls *.py
510  ls *.pdf
511  ls *.pdf
512  cp *.pdf demo/
513  cd demo
514  ll
515  ls
516  rm hoge.pdf
517  rm -f test.pdf
518  cd
519  rm -rf demo
520  history
~$ !518
cd
(reverse-i-search)`de': rm -rf demo
```

コマンドの履歴を検索できる。Ctrl+Rのあと「de」と入力したところ。

Emacs キーバインド

- ❖ Ctrl+F 右へ、Ctrl+B 左へ
- ❖ Ctrl+P 上へ、Ctrl+N 下へ
- ❖ Ctrl+A 行の先頭、Ctrl+E 行の最後
- ❖ Ctrl+D 右1文字を消す、Ctrl+H 左一文字を消す、Ctrl+K ここから最後まで消す
- ❖ Aの横をCtrlにすると、カーソルキーをほとんど使わなくてよくなる
 - ❖ macOSだと多くのアプリで、このキーバインドが使えます

標準入出力

```
tsujishingo — bash — 76x16
~$
~$ cat
test
test
~$
~$ cat < test.txt
test string
~$ cat test.txt
test string
~$
```

入力を待つので、「test」と入力

「test」と出力される

ファイルtest.txtから入力を受け取る

こちらでも同じ

標準エラー出力

標準入力から受け取った文字列を、標準出力に返す

```
temp = input()  
print(temp)
```

これも同じ

```
import sys  
temp = input()  
sys.stdout.write(temp)
```

標準エラー出力

```
import sys  
temp = input()  
sys.stderr.write(temp)
```

出力先の違い

```
import sys
```

```
sys.stdout.write('標準出力\n')
```

```
sys.stderr.write('標準エラー出力\n')
```

demo — bash — 76x16

```
~/demo$ python std_err.py
```

標準出力

両方向面にでる

標準エラー出力

```
~/demo$ python std_err.py > out.txt
```

標準エラー出力

> はリダイレクト。

```
~/demo$ cat out.txt
```

標準出力のみファイルに書き出される

標準出力

```
~/demo$ python std_err.py >> out.txt
```

標準エラー出力

>> は追加モード

```
~/demo$ cat out.txt
```

標準出力

相変わらず、標準エラー出力は画面

標準出力

```
~/demo$
```

時々使う 2>&1

demo — bash — 76x16

```
~/demo$
```

```
~/demo$ python std_err.py > out.txt 2>&1
```

```
~/demo$
```

```
~/demo$ cat out.txt
```

標準エラー出力

標準出力

```
~/demo$
```

> out.txt

標準出力はファイルへリダイレクト

2>&1

標準エラー出力（2番）を1番と同じ場所へ

Mac限定

- ❖ ターミナルから、「open .」でFinderが起動します
 - ❖ しかも、カレントディレクトリが開く
- ❖ Finderからファイルを投げ込むと、フルパスをターミナルの文字列として展開してくれます

bashの設定

- ❖ ホームディレクトリに.（ドット）から始まるファイルがある
- ❖ `ls -a`としないと通常、見えません
- ❖ 基本は`.bash_profile`、`.bashrc`
- ❖ `.bash_profile`がログイン時、`.bashrc`がシェルの起動時に読み込まれる

基本的な書き方

- ❖ export 変数名="値"
- ❖ \$変数名
 - ❖ 変数の中身を表示
- ❖ 区切りは「:」
- ❖ 結構ユルい（気がする）
- ❖ 追記したら、
 - ❖ source .bashrc

```
11 DISPLAY=:0.0
12 export DISPLAY
13
14 # Source global definitions
15 if [ -f /private/etc/profile ]; then
16     . /private/etc/profile
17 fi
18 if [ -f /private/etc/bashrc ]; then
19     . /private/etc/bashrc
20 fi
21 if [ -f /private/etc/profile-osxws ]; then
22     . /private/etc/profile-osxws
23 fi
24
25 ###
26 ### User specific aliases and functions
27 ###
28
29 #stty -ixon
30
31 # path setting
32 if ! echo $PATH | /usr/bin/grep -q "/usr/X11R6/bin"; then
33     PATH="/usr/X11R6/bin:${PATH}"
34 fi
35
36 if ! echo $PATH | /usr/bin/grep -q "/usr/local/sbin"; then
37     PATH="/usr/local/sbin:${PATH}"
38 fi
39
40 if ! echo $PATH | /usr/bin/grep -q "/usr/local/bin"; then
41     PATH="/usr/local/bin:${PATH}"
42 fi
43
44 if ! echo $PATH | /usr/bin/grep -q "${HOME}/bin"; then
45     PATH="${HOME}/bin:${PATH}"
46 fi
47
48 if ! echo $MANPATH | /usr/bin/grep -q "/usr/local/share/man"; then
49     MANPATH="/usr/local/share/man:/usr/share/man:/usr/X11R6/man"
50 fi
51 PATH="/usr/local/mysql/bin:/usr/local/teTeX/bin:/Developer/usr/bin:${PATH}"
52 MANPATH="/usr/local/teTeX/man:${MANPATH}"
53 export PATH MANPATH
54
55 # for Postgresql
56 export PATH=$PATH:/Applications/Postgres.app/Contents/Versions/latest/bin
57
58
59 # set LANG
60 #export LANG=ja_JP.eucJP
61 #export LC_ALL=ja_JP.eucJP
62 export LANG=ja_JP.UTF-8
63 export LC_ALL=ja_JP.UTF-8
64
65 # 'ls' color setting
66 #export LSCOLORS="ExFxCxDxBxEGEDABAGACAD"
67 export LSCOLORS=GxdxcxdxBxegedabagacad
```


コマンドサーチパス

```
demo — bash — 76x16
~/demo$
~/demo$ which ls    lsはどこにあるのか？
/bin/ls
~/demo$ echo $PATH
/Users/tsujishingo/anaconda3/bin:/Users/tsujishingo/lib:/anaconda/bin:/Library/Frameworks/Python.framework/Versions/3.4/bin:/usr/local/mysql/bin:/usr/local/teTeX/bin:/Developer/usr/bin:/Users/tsujishingo/bin:/usr/local/sbin:/usr/X11R6/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/Library/TeX/texbin:/opt/local/bin:/opt/local/sbin:/Applications/Postgres.app/Contents/Versions/latest/bin
~/demo$
```

環境変数PATHの中から探せる

環境変数

- ❖ env

- ❖ 設定されている環境変数が全部みられます

- ❖ conda create など Python の仮想環境を作る

- ❖ 実体は、環境変数を調整して、適切なバージョンの Python やライブラリが動くようにしている

ファイル処理

demo — bash — 76x16

```
~/demo$
```

```
~/demo$ cat data.txt
```

```
ID      name      age
```

```
1        たろう    22
```

```
2        はなこ    21
```

```
3        じろう    19
```

```
4        ようこ    20
```

```
~/demo$ head -2 data.txt
```

```
ID      name      age
```

```
1        たろう    22
```

```
~/demo$ tail -2 data.txt
```

```
3        じろう    19
```

```
4        ようこ    20
```

```
~/demo$
```

全体を表示

先頭2行

最後の2行

並べ替え

```
demo — bash — 76x16
~/demo$
~/demo$ sort data.txt
1      たろう  22
2      はなこ  21
3      じろう  19
4      ようこ  20
ID      name   age
~/demo$ sort -nk3 data.txt
ID      name   age
3      じろう  19
4      ようこ  20
2      はなこ  21
1      たろう  22
~/demo$
```

1行を1つの単語として、文字列としてソート

3列目を数字としてソート

grep とパイプ |

```
demo — bash — 76x16
~/demo$
~/demo$
~/demo$ grep "^[0-9].*" data.txt
1      たろう  22
2      はなこ  21
3      じろう  19
4      ようこ  20
~/demo$
~/demo$ grep "^[0-9].*" data.txt | sort -nrk3
1      たろう  22
2      はなこ  21
4      ようこ  20
3      じろう  19
~/demo$
```

grepで数字から始まる行だけ取り出す

パイプ | でsortに結果を渡す

awk

```
demo — bash — 76x16
~/demo$
~/demo$ awk '{print $2,$3}' data.txt
name age
たろう 22
はなこ 21
じろう 19
ようこ 20
~/demo$
~/demo$ awk 'BEGIN{OFS="\t"} {print $2,$3}' data.txt
name      age
たろう    22
はなこ    21
じろう    19
ようこ    20
~/demo$
```

2列目と3列目だけを表示

OFS(output field separator)をタブに設定し直す

UNIXの考え方

```
demo — bash — 76x16
~/demo$
~/demo$
~/demo$ cat num.txt
```

```
2
3
4
3
4
3
4
6
7
6
5
4
3
2
~/demo$ cat num.txt | sort
```

```
2
2
3
3
3
3
4
4
4
4
5
6
7
~/demo$ cat num.txt | sort | uniq
```

```
demo — bash — 76x16
~/demo$
~/demo$
~/demo$ cat num.txt | sort | uniq | wc -l
6
```

```
~/demo$
~/demo$
```

まとめ

- ❖ どこでも使えるbashを紹介
- ❖ ワイルドカード*を使ったファイルの処理
- ❖ 環境変数やaliasの設定
- ❖ 便利なEmacsキーバインド
- ❖ 小さなコマンドを組み合わせ、大きな仕事をする