FreeBSD をさわってみよう　　c シェル　3

プロセスの更新をリアルで見る。

```
% top
```

```
% top
last pid:   742; load averrages: 0.18,0.19, 0.00  up 0+00:03:52  04:33:50
16 processes: 1 running, 15 sleeping
CPU states:  0.4% user,  0.0% nice,  0.8% system, 0.4% interrput, 98.0% idle
Mem: 5748k Active, 3896k Inact, 8596k Wired, 7264k Buf, 67M Free
Swap: 136M Total, 136M Free
Sep 20 04:31:39 kernel: pid 742 (fsck_ufs), uid 0 inumber 2182 on /usr: filesys
tem fullERNAME  THR PRI NICE   SIZE     RES STATE    TIME   WCPU COMMAND
734 hiro          1   96    0  2324k  1540k  RUN       0:00  0.00%  top
676 root          1   96    0  3508k  2776k  select    0:01  0.00%  sendmail
732 hiro          1   20    0  1404k  1080k  select    0:00  0.00%  sysload
724 root          1    8    0  3700k  2692k  pause     0:00  0.00%  csh
552 root          1   96    0  1788k  1452k  wait      0:00  0.00%  login
688 root          1    8    0  1396k  1112k  nanslp    0:00  0.00%  cron
625 root          1   20    0  1296k   820k  select    0:00  0.00%  usbd
731 root          1    5    0  1352k   948k  ttyin     0:00  0.00%  getty
725 root          1    5    0  1352k   948k  ttyin     0:00  0.00%  getty
727 root          1    5    0  1352k   948k  ttyin     0:00  0.00%  getty

.
.
.       -------途中省略----------
```

終了する時は q を押す。

引数が無いと、自分のみのプロセスだけでなく、全てのプロセスが表示されてしまいます。

自分のプロセスを見てみましょう。

コマンド　top –U ユーザ名

```
top –U hiro
```

```
last pid:   967; load averrages: 0.00,0.00, 0.00  up 0+01:47:38  06:33:40
16 processes: 1 running, 15 sleeping
CPU states:  0.4% user,  0.0% nice,  0.8% system, 0.4% interrput, 98.0% idle
Mem: 5632k Active, 3992k Inact, 8620k Wired, 7264k Buf, 67M Free
Swap: 136M Total, 136M Free

tem fullERNAME  THR PRI NICE   SIZE     RES STATE    TIME   WCPU COMMAND
734 hiro          1   96    0  2300k  1516k  RUN       0:09  0.00%  top
724 root          1    8    0  3828k  2696k  pause     0:00  0.00%  csh
```

書式

top

プロセスの情報を表示する。
オプション
-U ユーザ名　：　ユーザ名が実行しているプロセスのみを表示する。
-s 秒：　画面の更新を秒で設定する。
-b　：　バッチモードにする。(非対話)
-i　：　インタラクティブモード　(対話)
-d 回数：　画面の更新回数だけ表示して、終了する。

バッチ処理 : **バッチ処理**とは、コンピュータで 1 つの流れのプログラム群（ジョブ）を順次に実行すること。まえまえから定めた処理を一度に行うこと
インタラクティブ : インタラクティブ（interactive）は「相互に作用する、対話形式の」といった意味を持つ。与えられた情報をただ受け取るだけでなく、それに対してユーザーから何らかの働きかけができる状態のこと

インタラクティブモード（対話）

q:　終了
d:　表示する回数を決めて、終了する。
n:　表示するプロセスの数を決める。
s:　更新する間隔を秒で決める。
k:　シグナルを　kill コマンドで、送る。
u:　ユーザ名を指定して、ユーザのみのプロセスを表示する。

表示リスト

PID　：　プロセス ID
USERNAME　：　プロセス所有者名
PRI　：　現在のプロセス優先度
NICE　：　nice 値
SIZE　：　プロセスサイズの合計
RES　：　現在のメモリ常駐量
STATE　：現在の状態
TIME　：　プロセスが消費したシステム時間およびユーザ時間の秒数
WCPU　：　CPU パーセンテージ加重 CPU 使用率(process の scheduling に使用)
CPU　：　生の CPU パーセンテージで、プロセス順序を決めるソート処理に用いられる(表示順)
COMMAND：プロセスが現在実行しているコマンド名
例
% top –U user

FreeBSD はマルチユーザ、マルチタスクシステムなので、同時に複数のプロセスを実行できるが、その際どのプロセスを優先的に実行するかを指定できる。

プライオリティ（NI(nice 値)）は -20 から 20 までの値を取る。
プライオリティ（NI(nice 値)）とはプロセスの優先度のことで、値が低いほどスケジューリングにおける優先度が高くなる。
つまり -20 が最も優先順位が高く、高速に動くわけである。逆に、重要でないプロセスの優先度を低く設定しておけば、他のプロセスが実行される機会が増えることになる。

（NI(nice 値)）は、nice 値を指定するか、指定をしないで、デフォルト（規定値）の nice 値　4 が増えます。

他の人に迷惑をかけない用に　重い計算をする時などに、使います。優先度を低くして、他の人の計算処理の資源に回します。これを、優先順位式スケジューリングという

優先順位式スケジューリング：実行可能なプロセスの中で、一番優先順位が高いプロセスを実行する

例
プロセスの優先順位を表示する為に、ps l コマンドを使います。

```
% ps l
UID   PID   PPID   CPU   PRI   NI    VSZ    RSS   MWCHAN   STAT   TT      TIME     COMMAND
1001   798     721     0    20     0    3700   2696   pause    S      v0      0:00.18   -csh (csh)
1001   815     798     2    96     0    1428   904    -        R+     v0      0:00.01    ps l
%
```

ps l コマンドを　+12 で優先順位を下げて実行します。

```
% nice +15 ps l

UID   PID   PPID   CPU   PRI   NI    VSZ    RSS   MWCHAN   STAT   TT      TIME     COMMAND
1001   798     721     1    20     0    3828   2700   pause    S      v0      0:00.22   -csh (csh)
1001   904     798     3   111    15    1428   904    -        RN+    v0      0:00.01    ps l
%
```

書式

nice [オプション]コマンド[引数]

オプション
nice 値

例

ps l コマンド　を　nice 値 7 に優先順位を下げて実行

% nice +7 ps l

注意
nice 値は、スーパーユーザだけが、優先順位を上げる事が出来る。
nice 値は、は、-20 から+20 までの範囲がある。
デフォルトのオプションのナイス値を書かないで実行すると、nice 値は、0 に+4 増加する。

何個か前に使用したコマンドをまた、使いたい時に、そのつど、入力するのは、面倒です。
以前に入力したコマンドを、参照するのに、history コマンドがあります。

history コマンドを、使ってみましょう。

まずは、ls コマンド、pwd コマンド、date コマンドを実行してみましょう。
次に、history コマンドを、使ってどのように表示されるかを、見てみます。

```
% ls
1               a               banana          oden            z.jpg
2               ame             c               purin           z.text
3               anko            d               yakisoba
4               b               hotdog          z.html
% pwd
/usr/home/hiro
% date
Thu Sep 25 14:06 :19 UTC 2008
%history
          1 13:43   h
          2 13:43   h
          3 13:44   history
          4 13:44   vi .cshrc
          5 13:46   ls
          6 13:46   histry
          7 13:46   h
          8 13:46   ls
          9 14:04   pwd
         10 14:06   date
%
```

表示されたコマンドの左端にヒストリー番号がついています。ヒストリー番号を「! ヒストリー番号」と入力すると、
番号のコマンドを実行した事になります。

例えば

```
% !9
pwc
/usr/home/hiro
%
```

次は、「!!」と入力すると直前に実行したコマンドを実行します。

```
% ls
1               a               banana          oden            z.jpg
2               ame             c               purin           z.text
3               anko            d               yakisoba
4               b               hotdog          z.html
% !!
ls
1               a               banana          oden            z.jpg
2               ame             c               purin           z.text
3               anko            d               yakisoba
4               b               hotdog          z.html
```

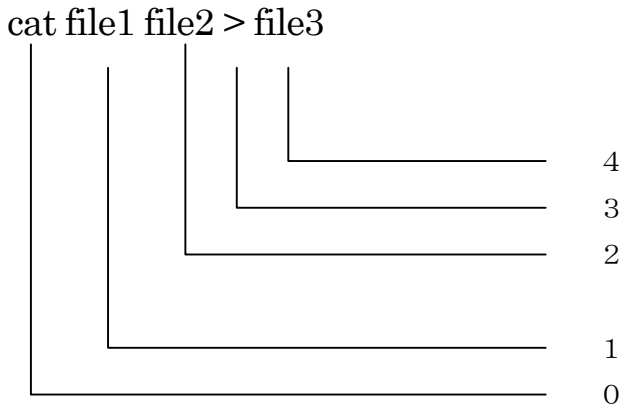
また、「!コマンドの頭部分」を実行すると、最近のコマンドの頭部分に該当する。コマンドを実行します。

```
% !pw
pwd
/usr/home/hiro
% !p
pwd
/usr/home/hiro
%
```

c シェルには、history コマンドで、表示された番号で、以前のコマンドを実行するだけでなく、以前のコマンドからの引数を、「!4」「!コマンドの頭部分」などを使って新しいコマンドラインを作る事ができます。
!4」「!コマンドの頭部分」の引数指定をして、その引数指定の後ろに、「:」をつけて、引数指定子と呼ばれる引数を指定する。

引数指定子は、コマンドラインを左端を　0(ゼロ)として空白で区切り、番号順に示したものです。

引数指定子の番号のふりかた



「>」「<」「|」「&」「>>」などの、ｃシェルが扱う特殊文字は、１つの単語として扱われる。

例
cat > kyakuA コマンドを使い　ls コマンドを使って kyakA を見てみます。

```
% cat > kyakuA
setagayaku                sakurai          wakaba
ko-to-ku                  suzuki           katumi
sumidaku                  mita             yoshiko
setagayaku                kura             toshiko
% ls –al !!:1
ls –al kyakuA
-rw-r—r--    1 hiro wheel    97 Sep 14 21:01 kyakuA
%
```

ヒストリー番号をつかい引数指定子の番号を指定する事が出来ます。

まずは、history コマンドを使い、　ls –al kyakuA のヒストリー番号を探して使ってみます。

ls –al kyakuA の今回のわたしのヒストリー番号が、78 ですが、皆さんのは同じでは、ないので注意してください。

```
% history
・
・
・  ―――途中省略―――

・
76   21:12    cat kyakuA
78   21:19   ls –al kyakuA
79   21:23   ls
80
% !78:0 !78:1 !78:2
ls –al kyakuA
-rw-r—r--   1 hiro wheel   97 Sep 14 21:01 kyakuA
%
```

つぎは、引数指定子の番号を１番から３番までと指定をして見ます。
まず、ls kyakuA kyakuB kyakuC とコマンドを打ちます。次に cat コマンドを使い、
cat !!:1-3 > kyakuall と、引数指定子の番号を１番から３番までと範囲を指定をして
ファイル kyakuA kyakuB kyakuC を連結して、kyakuall を作ります。
最後に kyakuall を削除します。

```
% ls kyakuA kyakuB kyakuC
kyakuA kyakuB kyakuC
% cat !!:1-3 > kyakuall
cat kyakuA kyakuB kyakuC > kyakuall
% ls kyakuall
kyakuall
% rm !!:1
rm kyakuall
% ls kyakuall
ls:kyakuall: No such file or directory
%
```

先程と同じように、
ls kyakuA kyakuB kyakuC　と打ち込み
cat !ls:1-3 >　kyakuall と打ち込みます。
先程との違いは、!ls　と　!!: の違いです。
そして、kyakuall を確認の後に削除します。

連続して
cat !ls:1-$ > kyakuall
これは、引数指定子を　1から最後まで指定するということです。すなわち 1-3 を指定した事とおなじです。確認して削除します。
また、同様に　ls kyakuA kyakuB kyakuC　と打ち込み
cat !ls: 1* > kyakuall と打ち込み、確認して削除します。

```
% ls kyakuA kyakuB kyakuC
kyakuA kyakuB kyakuC
% cat !ls:1-3 > kyakuall
cat kyakuA kyakuB kyakuC > kyakuall
% ls kyakuall
kyakuall
%rm !!:1
%
% ls kyakuA kyakuB kyakuC
kyakuA kyakuB kyakuC
% cat !ls:1-$ > kyakuall
cat kyakuA kyakuB kyakuC > kyakuall
% ls kyakuall
kyakuall
%rm !!:1
%
% ls kyakuA kyakuB kyakuC
kyakuA kyakuB kyakuC
% cat !ls:1* > kyakuall
cat kyakuA kyakuB kyakuC > kyakuall
% ls kyakuall
kyakuall
%rm !!:1
%
```

続きです。
直前の全てのコマンドの引数を指定するには、「!*」
直前の最後の引数を指定するには、「!$」を指定する。

```
% ls kyakuA kyakuB kyakuC
kyakuA kyakuB kyakuC
% ls !*
ls kyakuA kyakuB kyakuC
% ls !$
ls kyakuC
kyakuC
%
```

書式

history [オプション][表示する数]

オプション

-h 時間とヒストリー番号を表示しない
-r 時系列を逆に表示する。

例
表示できる。過去のコマンドと時間とヒストリー番号を表示する。
% history

直前のコマンドから１３個前までの history を表示する。
% history 13

ヒストリー番号 7 のコマンドを実行する。
% !7

直前のコマンドを実行する。
!!

エイリアス

コマンドに別名を付ける事が c シェルでは、できます。

alias 別名　コマンド
例

```
% alias looking pwd
% looking
/usr/home/hiro
%
```

現在 alias で、設定されているコマンドを知りたい時は、
alias と入力して Enter キーを押します。

```
%alias
h       (history 25)
j       (jobs –l)
la      (ls –a)
lf      (ls –FA)
ll      (ls –lA)
looking ls
%
```

エイリアスを設定しても、logout すると、設定は、消えてしまいます。
先程　alias コマンドを使って、自分では、設定していないのに、表示されたコマンドは、
FreeBSD で始めから設定されているコマンドです。exit コマンドを実行して、logout してから、もう一度　login をして、alias コマンドを実行してみてください。

```
% exit
login:
Password:
% alias
h       (history 25)
j       (jobs –l)
la      (ls –a)
lf      (ls –FA)
ll      (ls –lA)
%
```

c シェルでは、<,>, | 引数などが、使えます。

例

% alias lm 'ls | more'

alias に、登録されている。alias の別名のコマンドを利用する事もできます。

例
la　(ls –a)　で、定義された、la コマンド　を、また、別名 ss と言う別名で定義してみます。

```
% h
h        (history 25)
j        (jobs –l)
la       (ls –a)
lf       (ls –FA)
ll       (ls –lA)
looking  ls
% alias ss la
% ss
.                4               b               oden            z.text
..               ?               banana          purin
1                a               c               yakisoba
2                ame             d               z.html
3                anko            hotdog          z.jpg
%
```

エイリアスの設定を取り消すためには、
unalias コマンドを使います。

例

```
% unalias ss
% ss
ss:Command not found.
%
```

書式

aliasi [コマンドの別名] [コマンド列]

例

exit コマンドに、q というコマンドの別名をつける。
% alias q exit

date コマンドに　d というコマンドの別名をつける。
% alias d date

書式

unaliasi [コマンドの別名]

例

alias で、作った　q というコマンドの設定を、解除する。
% unalias q

alias で、作った　d というコマンドの設定を、解除する。
% unalias d

alias で、作った　q というコマンド　と　d というコマンドの設定を、一緒に解除する。
% unalias q d

Ctrl +D (Ctrl キーを押しながら d キーを押す)　と Ctrl +[（Ctrl キーを押しながら[キーを押す]　で、ファイル名補完

練習

ファイル名 purin を補完する。

p から、始まるファイル名は、purin だけです。

まずは、　ls p　まで、入力します。

```
% ls
1               a               banana          oden            z.jpg
2               ame             c               purin           z.text
3               anko            d               yakisoba
4               b               hotdog          z.html
% ls p
```

次に、Ctrl +[（Ctrl キーを押しながら[キーを押す]で、ファイル名を補完します。

```
% ls purin
```

練習

ファイル名 anko を補完する。

まずは、ls a　まで、入力する。

```
% ls a
```

次に、Ctrl +[（Ctrl キーを押しながら[キーを押す]で、ファイル名を補完します。

```
% ls a
```

補完されない

補完されないので、次は、Ctrl +D (Ctrl キーを押しながら d キーを押す)を押します。

```
% ls a
a*          ame*        anko*
% ls a
```

a*　　ame*　　anko* のように表示しましたら、

anko の次の文字の「n」を入力して、Ctrl +[（Ctrl キーを押しながら　[キーを押す）で、ファイル名を補完します。

```
% ls an
% ls anko
```

cat

cat　コマンドは、ファイルの中身をみるコマンドですが、下記のようにファイルを作ってみましょう。

ファイル名は　 aisatu1
　中身の文は　 ohayougozaimasu
　　aisatu1 は（あいさつ １ ）のことで　ohayougozaimasu は（ おはようございます ）のことです。

まず、　cat > aisatu1　と文字を打ってみましょう。

```
% cat > aisatu1
```

aisatu1 の文字をうちこみましたら、 Enter キーを押します。

次に、中身の文の　ohayougozaimasu の文字を入力して、 Enter キーを押します。

```
% cat > aisatu1
  ohayougozaimasu
  ■
```

の画面になりましたら

Ctrl + D　（コントロールキーを押しながら d を押す）を行います。

```
%cat > aisatu1
  ohayougozaimasu
  % D
```

上記の様に表示されます。

次に、 aisatu1 と言うファイルがあるかを確認します。

```
% ls
aisatu1
```

ファイル名　aisatu1　を確認できましたら、 ファイルの中身を見ましょう。

cat aisatu1　と入力して Enter キーを押します。

```
% cat aisatu1
   ohayougozaimasu
%
```

　ファイル名　aisatu1　の中身　ohayougozaimasu　が見れましたら、成功です。

ｃat　コマンドは、テキストファイルの中身を見る事ができるのですが、
バイナリファイルは、テキストファイルではないので、中身を見ようとすると、訳が解らない文字列が表示されてしまいます。

バイナリファイル：　(binary file) 専用のアプリケーションソフトで扱うことを前提として、文字コードの範囲などを考慮せずに作成されたファイルのこと。画像や動画、音声を記録したファイルや、実行可能形式のプログラムを収めたファイルなど、文字のみで構成されるテキストファイル以外はすべてこれに含まれる

次は

ファイル名は　aisatu2
　中身の文は　yorosikuonegaisimasu
　　aisatu2 は（あいさつ 2 ）のことで　yorosikuonegaisimasu は（ よろしくおねがいします ）のことです。

まず、　cat > aisatu2　と文字を打ってみましょう。

```
% cat > aisatu2
```

aisatu2 の文字をうちこみましたら、 Enter キーを押します。

次に、中身の文の　yorosikuonegaisimasu の文字を入力して、 Enter キーを押します。

```
% cat > aisatu2
  yorosikuonegaisimasu
  ■
```

の画面になりましたら

Ctrl + D　（コントロールキーを押しながら d を押す）を行います。

```
% cat > aisatu2
  yorosikuonegaisimasu
  % D
```

上記の様に表示されます。

次に、 aisatu1 と言うファイルがあるかを確認します。

```
% ls
aisatu1　　aisatu2
```

ファイル名　aisatu2　を確認できましたら、 ファイルの中身を見ましょう。

cat aisatu2　と入力して Enter キーを押します。

```
% cat aisatu2
   yorosikuonegaisimasu
%
```

ファイル名　aisatu2　の中身　yorosikuonegaisimasu　が見れましたら、成功です

次に、 2 つのファイル　aisatu1　と　aisatu2　のファイルを同時に、 見てみましょう。

```
% cat aisatu1 aisatu2
ohayougozaimasu
yorosikuonegaisimasu
%
```

cat　の、コマンド名は　（　concatenate　連結する ）の略です。

c at　コマンドは、 1 画面しか表示できません。そんな時は、 cat　 の代わりに
more　コマンドをつかいます。

more /COPYRIGHT　と入力してみましょう。

```
%more　/COPYRIGHT
```

すると、画面いっぱいに英語で、一画面では表示しきれずにいます。
Enter キーを押すと 1 行ずつ下に移動します。矢印キー ↓ ↑ でも 移動できます。

cat

cat　コマンドは、ファイルの中身をみるコマンドですが、下記のようにファイルを作ってみましょう。

ファイル名は　 aisatu1
　中身の文は　 ohayougozaimasu
　　aisatu1 は（あいさつ 1 ）のことで　ohayougozaimasu は（ おはようございます ）のことです。

まず、　cat > aisatu1　と文字を打ってみましょう。

```
% cat > aisatu1
```

aisatu1 の文字をうちこみましたら、 Enter キーを押します。

次に、中身の文の　ohayougozaimasu の文字を入力して、 Enter キーを押します。

```
% cat > aisatu1
  ohayougozaimasu
  ■
```

の画面になりましたら

Ctrl + D　（コントロールキーを押しながら d を押す）を行います。

```
%cat > aisatu1
   ohayougozaimasu
   % D
```

上記の様に表示されます。

次に、 aisatu1 と言うファイルがあるかを確認します。

```
% ls
aisatu1
```

ファイル名　aisatu1　を確認できましたら、 ファイルの中身を見ましょう。

cat aisatu1　と入力して Enter キーを押します。

```
% cat aisatu1
    ohayougozaimasu
%
```

　ファイル名　aisatu1　の中身　ohayougozaimasu　が見れましたら、成功です。

ｃ at　コマンドは、テキストファイルの中身を見る事ができるのですが、
バイナリファイルは、テキストファイルではないので、中身を見ようとすると、訳が解らない文字列が表示されてしまいます。

空白文字 Ent かタブ文字なのかを確認したい時は、オプション-t を付けて cat -vt コマンドを使い、
見えない文字を表示し、-v でを　タブを^I　と表示します。空白文字はそのままです

空白文字　＝　空
タブ文字　＝　田
Enter　＝　Ent

例

cat コマンドを使って，ファイル fff を作ります.

こんにちは　田　だんだんと 空 寒く 空 なりました 空 ね Ent
このごろは、空空空空空空空　景気　空　も　空　よろしく　空　ないので Ent
田田田テスト空空空空空テスト田田 Ent

konnitiha　　dandanto samuku narimasita ne
konogoroha　　keiki mo yorosiku nainode
Ent
Ent
END Ent

---

konnitiha 田 dandanto 空 samuku 空 narimasita 空 ne Ent
konogoroha 空空空空空空空空 keiki 空 mo 空 yorosiku 空 nainode Ent
田田田 tesuto 空空空空空 tesuto 田田 Ent
Ent
Ent
END Ent

```
% cat > fff
konnitiha　　dandanto samuku narimasitane
konogoroha　　keiki mo yorosiku nainode
　　tesuto　　tesuto


END
% cat –vt fff
konnitiha^Idandanto samuku narimasitane
konogoroha　　keiki mo yorosiku nainode
^I^I^Itesuto　　tesuto^I^I


END
%
```

次は、ファイル fff の改行文字　を表示する為に　オプション　–e を付けて表示します。
そして、オプション-n を使って行番号を表示しながら、見えない文字のタブと改行文字を表示します。
確認の為に wc コマンドを使います。

```
% cat –ev fff
konnitiha             dandanto samuku narimasita ne$
konogoroha          keiki mo yorosiku nainode$
                                tesuto        tesuto              $
$
$
END
% cat –nevt
             1   konnitiha^Idandanto samuku narimasita ne$
             2   konogoroha          keiki mo yorosiku nainode$
             3   ^I^I^Itesuto       tesuto^i^i$
             4   $
             5   $
             6   END$
% wc fff
        6        13         113 fff
%
```

cat コマンドは、データを連結して表示する事によく使われます。

ファイル fffcat1 と fffcat2 を連結して、fffcat3 をつくります。

ファイル fffcat1 には、今日は何曜日かカレンダーを見てみたら？
kyouha nanyoubi ka karenda- wo mitemitara?

ファイル fffcat2 には、date コマンドを使って みて見るよ
date komando wo tukatte mitemiruyo
を入力する。

```
% cat > fffcat1
kyouha nanyoubi ka karenda- wo mitemitara?
% cat > fffcat2
date komando wo tukatte mitemiruyo
% cat fffcat1 fffcat2 > fffcat3
% cat fffcat3
kyouha nanyoubi ka karenda- wo mitemitara?
date komando wo tukatte mitemiruyo
%
```

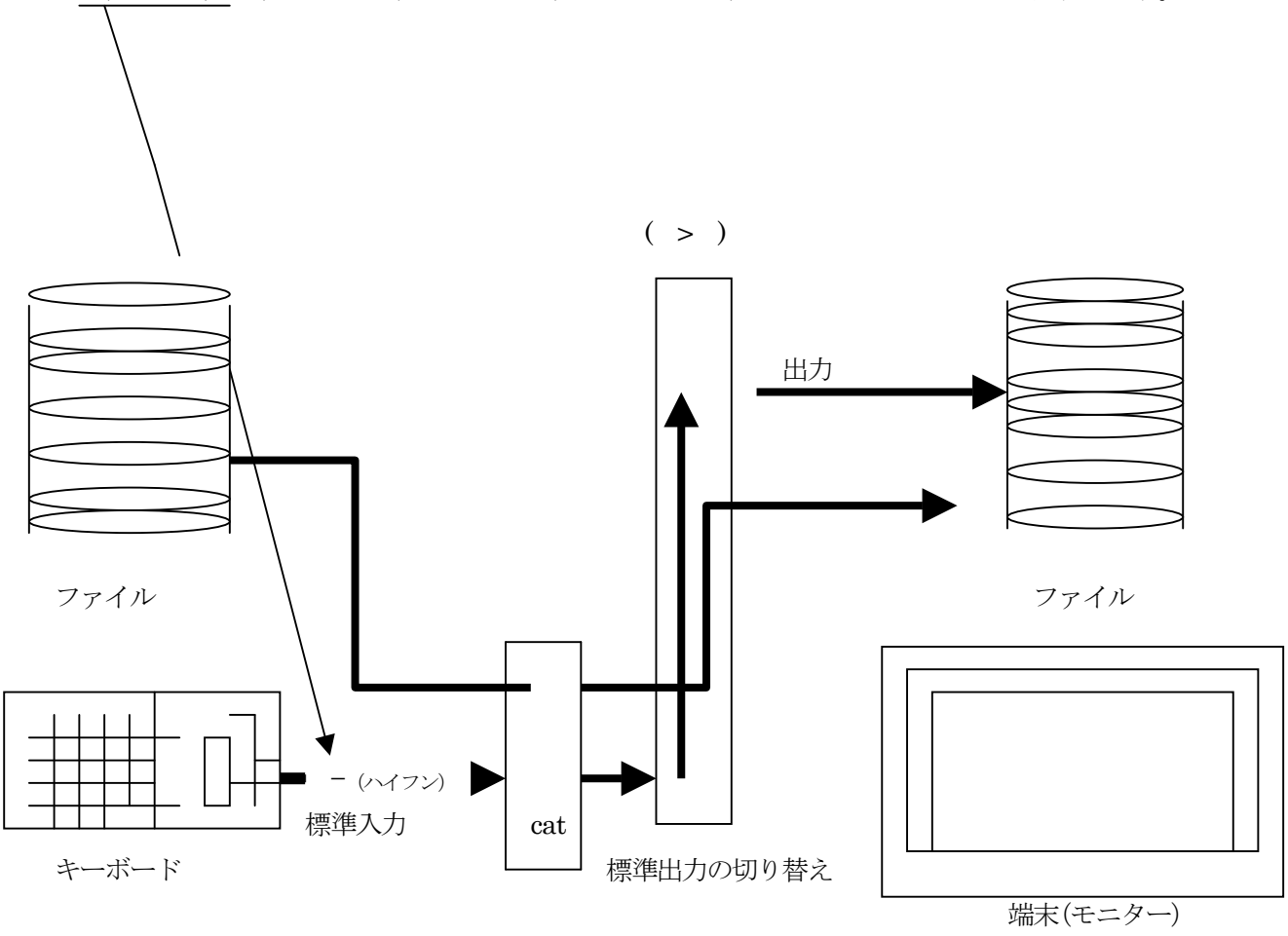
次に　存在しないファイル fffnot を fffcat1, fffcat2 とともに連結して、fffcat4 をつくってみます。

```
% cat fffcat1 fffcat2 fffnot > fffcat4
%　 cat fffcat4
cat: fffnot: No such file or directory
% cat fffcat4
kyouha nanyoubi ka karenda- wo mitemitara?
date komando wo tukatte mitemiruyo
%
```

cat: fffnot: No such file or directory　と　エラーメッセージが出てきますが、fffcat1, fffcat2 とともに連結した結果がファイル fffcat4 にきちんと記録され、エラーメッセージが記録されない事を、とどめておいてください。

標準入力から入力されるデータとファイルから出力されるデータが連結されて、標準出力の切り替えで
ファイルに出力するには、

cat －(ハイフン) ファイル1・ファイル2... ファイルｎ＞　ファイルA　コマンドを実行します。



例
はじめに、cat コマンドを使いファイル fffcatH を作ります。
そして、中身を確認後　cat－fffcatH＞fffcatJ コマンドで，ファイル fffcat J をつくります。

```
% cat > fffcatH
abcdefgh
% cat fffcatH
abcdefgh
% cat – fffcatH > fffcatJ
12345
% cat fffcatJ
12345
abcdefgh
%
```

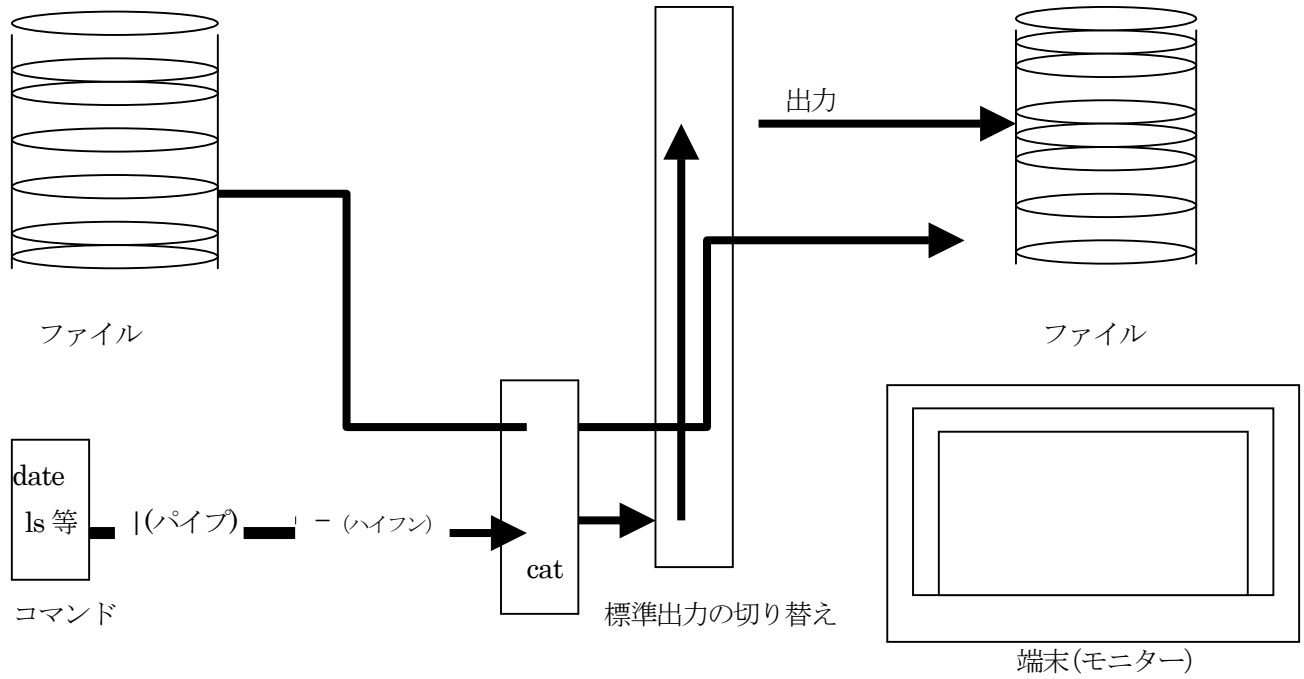
－（ハイフン）は、通常　標準入力、または、標準出力を表します。
先程の　% cat – fffcatH > fffcatJ　の　－（ハイフン）と　fffcatH を逆にして
cat fffcatH - > fffcatK と実行してみます。そして、出力の結果を比べてみてください。

```
% cat fffcatJ
12345
abcdefgh
% cat fffcatH - > fffcatK
12345
% cat fffcatK
abcdefgh
12345
%
```

12345 t と　abcdefgh　が逆になった事を確認してください。

コマンド　｜　ファイル　>　から　ファイルに出力される

（　>　）



コマンドからパイプにつながれて入力されるデータとファイルから出力されるデータが連結されて、標準出力の切り替えでファイルに出力するには、

コマンド　|（パイプ）　cat　－（ハイフン）ファイル１　ファイル２...ファイルｎ　＞　ファイルA

例
date コマンドの出力を先程のファイル fffcatH と cat コマンドで連結して、ファイル fffcatK をつくります。

```
% date | cat – fffcatH > fffcatK
Mon Oct 13 03:20:38 UTC 2008
abcdegh
%
```

次は、cat ファイル A ＞ ファイル A とコマンドを実行すると、出力する前のファイル A の中身は、出力後のファイル A では、消えてしまいます。

例
cat コマンドでファイル ggg をつくり,中身を確認してから cat ggg > ggg コマンドを実行して、
ファイル ggg の中身をみて、中身が無い事を確認します。
次に、ファイル sss をつくり、先程のファイル fffcatH と cat コマンドで連結して、
cat sss fffcatH > sss と実行した後、ファイル sss を作り、cat コマンドでファイ sss の中身をみる。

```
% cat fffcatH
abcdefgh
% cat > ggg
ggg
% cat ggg
ggg
% cat ggg > ggg
% cat ggg
%
% cat > sss
sss
% cat sss
sss
% cat sss fffcatH > sss
abcdefgh
%
```

出力後のファイル sss の中身は、出力前の sss の中身が消えて fffcatH の中身だけが残っています。

cat[オプション] [ファイル １　ファイル２・・・ファイル n]

オプション
-n　表示する内容に行番号をつける。
-b　表示する内容に行番号をつけるが、空白の行は、カウントしない。
-v　画面に表示出来ない文字コードを識別できるように出力する。
-e　行の終わりに $ を出力
-t　Tab(タブ文字)を^I として出力する。

例
ファイル　aisatu1 の内容を表示
% cat aisatu1

ファイル　aisatu1 aisatu2 aisatu3　を連結して表示
% cat aisatu1 aisatu2 aisatu3

標準入力した内容を表示
% cat
abc　　(と入力して　Enter キーを押すと)
abc　　(と表示される。)
orange　(と入力して　Enter キーを押すと)
orange　(と表示される。)

Ctrl + D(Ctrl　キーを押しながら d を押す。) をすると、終了です。

リダイレクトで、ファイル aisatu4 をつくる。
% cat　> aisatu4　　(Enter キーを押す。)
hajimemasite　　(Enter キーを押す。)
Ctrl + D(Ctrl　キーを押しながら d を押す。) をすると、ファイル　aisatu4 をつくります。
↓
% cat aisatu4
hajimemasite
%

標準入力から入力されるデータとファイル ffX のデータを連結してファイル ffBB をつくる。
% cat – ffX > ffBB

date コマンドの出力を先程のファイル fffcatH と cat コマンドで連結して、ファイル fffcatK をつくります。
% date | cat – fffcatH > fffcatK