レポートその７

以下の各問いに例を交え答えなさい．(参考：教科書 15 章)
次の知っていると便利なコマンドについて，それぞれコマンドの意味と使い方を例を挙げて書きなさい．

※例は実際に自分のターミナルに入れて表示されたものを記述します。

＜whatis＞

コマンドの名前からそのコマンドが何をするためのコマンドかを知るために使用します。man コマンドより簡略的な文章でそのコマンドの説明をします。

例

```
[N-juuuun-book:~] e095745% whatis which
```

と入力すると、画面が代わり、

```
which(1)                  - locate a program file in the user's path
```

と表示されます。

＜which＞

which コマンドは、コマンンド名からコマンドの本体がどこにあるかを調べるコマンドです。結果は絶対パス名で表示されます。

例

```
[N-juuuun-book:~] e095745% which mkdir
/bin/mkdir
[N-juuuun-book:~] e095745% which cat
/bin/cat
[N-juuuun-book:~] e095745% which man
/usr/bin/man
```

＜whereis＞

指定したコマンドの実行形式とソースプログラムファイル、およびリファレンスマニュアルのファイルの格納されているディレクトリを検索して、そのパス名を表示します。whereis は、コマンドサーチパスを探すのではなく、あらかじめ決められたディレクトリすべて探します。存在しないパス名を入力した場合は一つも表示されません。

例

```
[N-juuuun-book:~] e095745% whereis cat
/bin/cat
[N-juuuun-book:~] e095745% whereis mkdir
/bin/mkdir
[N-juuuun-book:~] e095745% whereis man
/usr/bin/man
[N-juuuun-book:~] e095745% whereis kkk
[N-juuuun-book:~] e095745% whereis cd
/usr/bin/cd
```

＜locate＞

ファイル名の一部を与えると絶対パスを探すコマンドです。

例

```
[N-juuuun-book:~] e095745% locate kadai2
/Users/e095745/prog1/kadai2
/Users/e095745/prog1/kadai2/#keisan002#
/Users/e095745/prog1/kadai2/#kesan003#
/Users/e095745/prog1/kadai2/a.out
/Users/e095745/prog1/kadai2/jougen
/Users/e095745/prog1/kadai2/jougen.c
/Users/e095745/prog1/kadai2/jougen.c~
/Users/e095745/prog1/kadai2/keisan.c
/Users/e095745/prog1/kadai2/keisan.c~
/Users/e095745/prog1/kadai2/keisan001
/Users/e095745/prog1/kadai2/keisan002
/Users/e095745/prog1/kadai2/keisan002.c
/Users/e095745/prog1/kadai2/keisan002.c~
/Users/e095745/prog1/kadai2/keisan002~
/Users/e095745/prog1/kadai2/kesan003
/Users/e095745/prog1/kadai2/kesan003.c
/Users/e095745/prog1/kadai2/kesan003.c~
/Users/e095745/prog1/kadai2/reibun
/Users/e095745/prog1/kadai2/reibun/balance
/Users/e095745/prog1/kadai2/reibun/reibun
/Users/e095745/prog1/kadai2/reibun/reibun.c
```

＜du＞

ユーザのディレクトリやファイルの大きさを調べるには du コマンドを使います。デフォルトではカレントワーキングディレクトリの下のディレクトリやファイルの大きさを調べます。du コマンドだけだとカレントワーキングディレクトリの下のすべてを調べてしまうので、%du パス名　で指定することもできます。

例
プログラミングで作ったファイルを入れているディレクトリ prog1 を du コマンドで大きさを調べてみました。

```
[N-juuuun-book:~/prog1] e095745% pwd
/Users/e095745/prog1
[N-juuuun-book:~/prog1] e095745% du
0    ./dir1
0    ./Hello
56   ./HelloWorld
256  ./kadai1/プログラ１　課題１
720  ./kadai1
72   ./kadai2/reibun
368  ./kadai2
304  ./kadai3
320  ./kadai4
16   ./kadai5/hallo.dSYM/Contents/Resources/DWARF
16   ./kadai5/hallo.dSYM/Contents/Resources
24   ./kadai5/hallo.dSYM/Contents
24   ./kadai5/hallo.dSYM
256  ./kadai5
0    ./man
240  ./waiwai
2280 .
[N-juuuun-book:~/prog1] e095745% du ./kadai3/
304  ./kadai3/
```

＜cmp＞
２つのファイルが同じかどうか調べるために使うコマンドです。２つのファイルが完全に一致している時は何も表示しません。

例
text01.c という”wahaha”と表示させるだけの簡単なプログラムを作って、試した結果を載せます。
text01 を cp コマンドでコピーしたものを text02 にします。text03 は”wahaha”を三回表示させるプログラムです。text04 は”DoraeMon”と表示させます。
text05 は text01 の後ろ二行を削除したものです。

```
[N-juuuun-book:~/prog1/man] e095745% cmp text01.c text02.c
[N-juuuun-book:~/prog1/man] e095745% cmp text01.c text03.c
text01.c text03.c differ: char 52, line 5
[N-juuuun-book:~/prog1/man] e095745% cmp text01.c text04.c
text01.c text04.c differ: char 38, line 4
[N-juuuun-book:~/prog1/man] e095745% cmp text01.c text05.c
```

```
cmp: EOF on text05.c
```

＜diff＞

２つのテキストファイルの異なる部分を探したい時は diff コマンドをつかいます。
diff は行単位で２つのファイルの異なる部分を表示します。

例
cmp コマンドで使用した text01～０５ファイルをで検証しようと思います。

```
[N-juuuun-book:~/prog1/man] e095745% diff text01.c text02.c
[N-juuuun-book:~/prog1/man] e095745% diff text01.c text03.c
4a5,6
>   printf("wahaha\n");
>   printf("wahaha\n");
[N-juuuun-book:~/prog1/man] e095745% diff text01.c text04.c
4c4
<   printf("wahaha\n");
---
>   printf("DoraeMon\n");
[N-juuuun-book:~/prog1/man] e095745% diff text01.c text05.c
5,6d4
<   return(0);
< }
```

＜comm＞

ソートされたファイルを行単位で比較して、共通な行と一方のファイルにしかない行に分類して表示するものです。なお、ファイルはあらかじめソートしておいてください。

例

```
[N-juuuun-book:~/prog1/man] e095745% comm text01.c text02.c
           #include<stdio.h>

           main(){
             printf("wahaha\n");
             return(0);
           }
[N-juuuun-book:~/prog1/man] e095745% comm text01.c text03.c
           #include<stdio.h>

           main(){
             printf("wahaha\n");
       printf("wahaha\n");
       printf("wahaha\n");
             return(0);
           }
[N-juuuun-book:~/prog1/man] e095745% comm text01.c text04.c
           #include<stdio.h>
```

```
          main(){
       printf("DoraeMon\n");
 printf("wahaha\n");
            return(0);
          }
[N-juuuun-book:~/prog1/man] e095745% comm text01.c text05.c
          #include<stdio.h>

          main(){
            printf("wahaha\n");
 return(0);
}
```

＜head＞

ファイルの先頭数行を表示するコマンドです。

例

以前にプログラミングの授業で作った ori_cryp_002.c というファイルでコマンドを試してみます。下はそのソースです。

```
[N-juuuun-book:~/prog1/kadai4] e095745% cat -n ori_cryp_002.c
  1 #include <stdio.h>
  2
  3 /*プロトタイプ宣言*/
  4 char rot( char c );
  5
  6 int main(void)
  7 {
  8   /*入力した文字を格納する変数の宣言*/
  9   char g;
 10
 11   /*EOF でループを抜け出す*/
 12   while( (g=getchar()) != EOF )
 13     {
 14       putchar( rot(g) );
 15     }
 16   return(0);
 17 }
 18
 19 /*復号化*/
 20 char rot( char c )
 21 {
 22   if (('!'<= c && c <='#')||('%'<= c && c <=39)||
```

```
 23       (')'<= c && c <='-')||('/'<= c && c <='3')||
 24       ('5'<= c && c <='9'))
 25     {
 26       return  ( c+33 );
 27     }
 28   if (('A'<= c && c <='C')||('E'<= c && c <='G')||
 29       ('I'<= c && c <='M')||('O'<= c && c <='S')||
 30       ('U'<= c && c <='Y'))
 31     {
 32       return  ( c+33 );
 33     }
 34   if(c == 'z')
 35     {
 36       return(97);
 37     }
 38   if(c == '{')
 39     {
 40       return(105);
 41     }
 42   if(c == '|')
 43     {
 44       return(117);
 45     }
 46   if(c == '}')
 47     {
 48       return(101);
 49     }
 50   if(c == '~')
 51     {
 52       return(111);
 53     }
 54
 55   if(91<= c && c <=100)
 56     {
 57       return(c-43);
 58     }
 59       return c;
 60     }
[N-juuuun-book:~/prog1/kadai4] e095745% head ori_cryp_002.c
  1. #include <stdio.h>
```

```
2.
3. /*プロトタイプ宣言*/
4. char rot( char c );
5.
6. int main(void)
7. {
8.  /*入力した文字を格納する変数の宣言*/
9. char g;
10.
```

この結果から head コマンドで空白の行も含めて、上から１０行表示されたのがわかります。また%head -行数 ファイル名　で表示する行数を指定することもできます。

＜tail＞
ファイルの末尾数行を表示するコマンドです。

例
head コマンドで使った同様のソースコードで試してみます。

```
[N-juuuun-book:~/prog1/kadai4] e095745% tail ori_cryp_002.c
1. {
2.    return(111);
3.   }
4.
5.  if(91<= c && c <=100)
6.   {
7.    return(c-43);
8.   }
9.    return c;
10.       }
```

この結果から tail コマンドで下から１０行表示されたのがわかります。また%tail +行数 ファイル名　+行数目以降の行を表示することもできます。

```
[N-juuuun-book:~/prog1/kadai4] e095745% tail +45
ori_cryp_002.c
```

```
  }
if(c == '}')
  {
   return(101);
  }
if(c == '~')
  {
   return(111);
  }

     if(91<= c && c <=100)
      {
       return(c-43);
      }
       return c;
    }
```

＜wc＞
ファイルの行数を知りたいときに使用するコマンドです。

例
```
[N-juuuun-book:~/prog1/kadai4] e095745% wc ori_cryp_002.c
   60     137     949 ori_cryp_002.c
```
左から「ファイルの行数」「単語数」「文字数」です。

＜split＞
ファイルを分割したい時は、split コマンドを使います。ファイルを分割して、指定された行数ごとに独立のファイルとして格納します。分割された時に作られるファイルの名前は、でふぉるとでは「x」のあとに「aa、ab、ac…az、ba、bb…」を辞書順に付けたものになります。また、デフォルトでは１０００行づつに分割しますが、これはエディタで読み込めないような大きなファイルを、分割して読み込めるようにするために作られたコマンドですので、とても大きい数になっているです。

例
```
[N-juuuun-book:~/prog1/man] e095745% split -20 ori_cryp_002.c
[N-juuuun-book:~/prog1/man] e095745% wc -l xa?
```

```
   20 xaa
   20 xab
   20 xac
   60 total
```

＜cat＞

引数で指定されたファイルの内容を順に標準出力に出力します。これを使って複数のファイルを結合できます。

例

spilit コマンドで分割したファイルを cat コマンドで結合してそのファイルが一致するかどうかやってみました。

```
[N-juuuun-book:~/prog1/man] e095745% cat xa? > link
[N-juuuun-book:~/prog1/man] e095745% ls
a.out          text01.c text03.c text04.c~xaa
link      text01.c~text03.c~text05.c xab
ori_cryp_002.c     text02.c text04.c text05.c~xac
[N-juuuun-book:~/prog1/man] e095745% cmp ori_cryp_002.c  link
```

＜script＞

画面に表示された実行結果をファイルに格納できます。これにより、実行結果にコメントを書き込んでプリントするというようなことができます。表示結果を記録するファイル名は指定できますが、デフォルトではカレントワーキングディレクトリの typescript という名前のファイルに記録します。script コマンドは新たにシェルを起動します。exit コマンドなどで終了すると、script も終了します。

例

comment というファイル名で script のファイルを作ってみました。

```
[N-juuuun-book:~/prog1/man] e095745% cat comment
Script started on Sun Jul  5 01:58:19 2009
bash-3.2$ cal
   7 月 2009
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 31

bash-3.2$ whatis script
```

```
bash-3.2$ exit
exit

Script done on Sun Jul  5 01:58:53 2009
```

プログラミングで実行結果を載せるときに役に立つと思ったので活用しようと思いました。

＜ページャとはなにか？＞
１画面に収まりきらないファイルを見るときに cat をつかうと、流れてしまって最初の部分が見れません。これは不便なのですが、かといってエディタを使うほどではない、というときに便利なのがページャです。ページャとは、ファイルの内容を１画面分ずつ表示するソフトウェアの総称です。表示する範囲は対話的に移動できます。

代表的なページャを次に３つ挙げる．それぞれ使い分けのための特徴を記せ.
＜more＞
more は非常に原始的なページャで、もともと表示領域を前方向に進めることしかできなかった。しかし、最近の実装では後方向へのスクロールも可能になっている。
＜less＞
more に似ているが、前方向だけでなく後方向にもスクロールできるよう拡張されている。
＜lv＞
UTF-8 で書かれた文章も含めて、日本語文章を一旦変換することなく表示させることができます。

＜ハードリンクとは何か説明しなさい．　また，使い方を例を挙げて書きなさい＞
まずリンクとは？
windows でいうショートカットのようなもので、unix にはハードリンクとシンボリックリンクの２つがあります。１つのファイルに複数のリンクを作成でき、リンク自体にはファイルの情報がある訳ではなく、ファイルの設置場所が記録されています。

ハードリンクはファイルの本名です。どんなファイルでも、１つ以上のハードリンクを持っています。すべてのハードリンクがなくなるとき、ファイルの実態が消されます。逆に言えばファイル名をすべて消さない限りファイルが消えることはありません。

例

```
[N-juuuun-book:~/prog1/link] e095745% ls
a.out
[N-juuuun-book:~/prog1/link] e095745% ln a.out link_h
[N-juuuun-book:~/prog1/link] e095745% ln a.out link_h2
[N-juuuun-book:~/prog1/link] e095745% ln a.out link_h3
[N-juuuun-book:~/prog1/link] e095745% ls -l
total 128
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 a.out
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 link_h
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 link_h2
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 link_h3
```

ハードリンクを３つ作ったので a.out と同じ名前を持ったファイルが４つできているのがわかる。

＜シンボリックリンクとは何か説明しなさい．　また，使い方を例を挙げて書きなさい＞
ハードリンクとは違い実体を移動してしまうとリンクが無効になってしまいます。ファイルの本体の位置情報を保管しているファイルで、ファイルの本体とは完全に区別されています。このため、ファイル本体がなくなった場合、参照先が見つからずエラーとなります。

例

```
[N-juuuun-book:~/prog1/link] e095745% ln -s a.out link_s
[N-juuuun-book:~/prog1/link] e095745% ln -s a.out link_s2
[N-juuuun-book:~/prog1/link] e095745% ln -s a.out link_s23
[N-juuuun-book:~/prog1/link] e095745% ls -l
total 152
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 a.out
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 link_h
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 link_h2
-rwxr-xr-x  4 e095745  staff  12588  4 21 16:02 link_h3
lrwxr-xr-x  1 e095745  staff      5  7  8 09:03 link_s -> a.out
lrwxr-xr-x  1 e095745  staff      5  7  8 09:03 link_s2 -> a.out
lrwxr-xr-x  1 e095745  staff      5  7  8 09:03 link_s23 -> a.out
```

＜感想＞
ページャがあまりよくわからなかった。下巻も買わないといけないと思った。

＜参考文献＞
新　THE UNIX Super Text 上　技術評論社
Unix コマンド　http://www.k-tanaka.net/unix/#u1