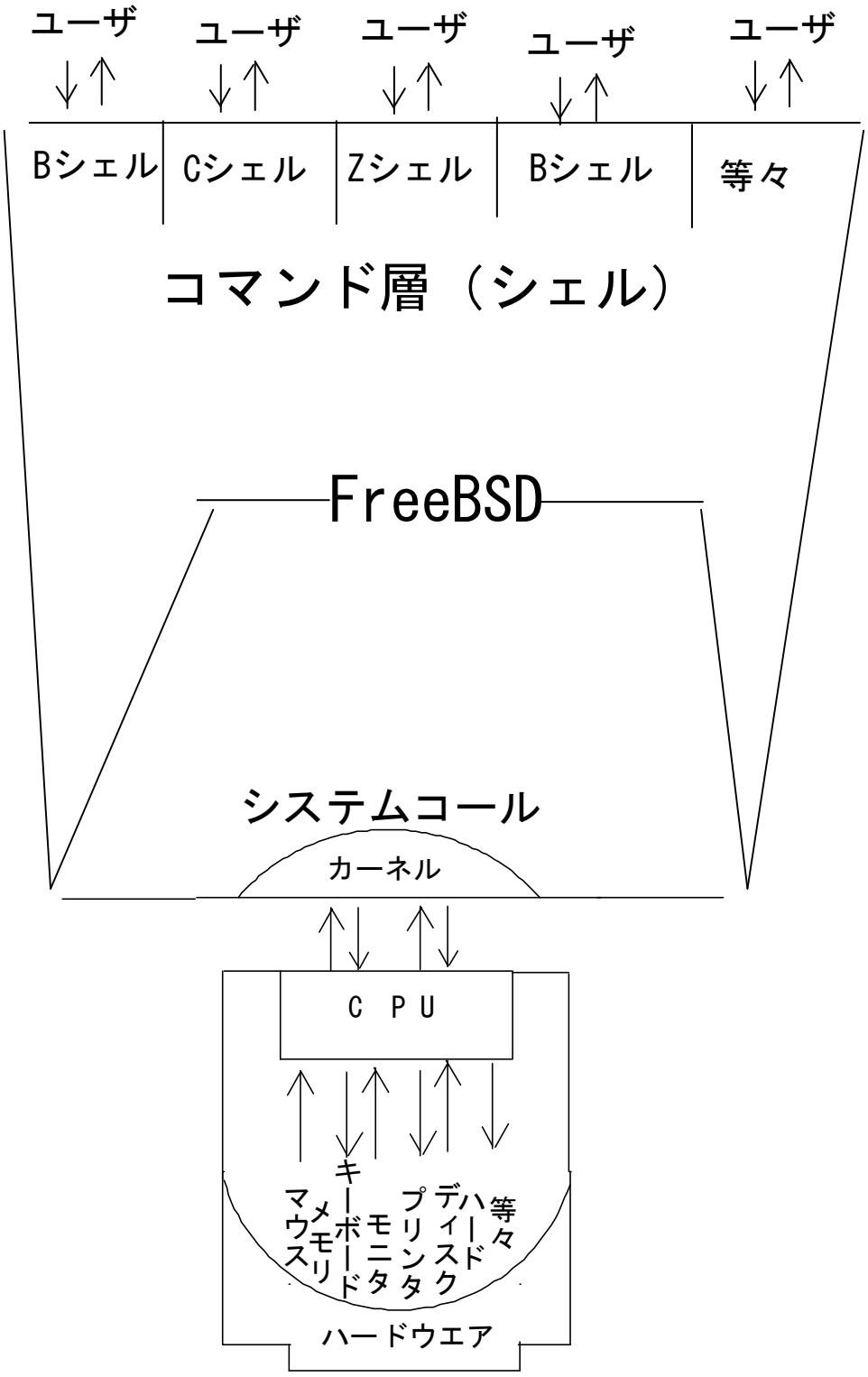
FreeBSD をさわってみよう　　ｃシェル１



先の図のように、

ユーザ
↓↑
シェル（FreeBSD）
↓↑
システムコール（FreeBSD）
↓↑
カーネル（FreeBSD）
↓↑
ハードウエア
の形で FreeBSD を利用しています。

# カーネル

Kernel とは、オペレーティングシステム(OS) の中核となる部分である。システムのリソース（CPU やメモリなど）を管理し、ハードウェアとソフトウェアの一体としてやりとりを管理する。
メモリ、CPU（中央処理装置）、などの入コンピュータのシステムの資源の入出力を中心としたハードウェアをファイルとして抽象化し、ハードウェアとソフトウェアがファイルとしてやり取りできるように管理する。
カーネルの機能は、CPU やメモリの管理だけではなく、他のプログラムがコンピュータのシステムの資源を動作出来るようにする事です。コンピュータの中核は、CPU または、マイクロプロッセサーです、その CPU または、マイクロプロッセサーがいろいろなプログラムをカーネルの管理下で、操作します。
次に大事な資源はメモリです。実行するプログラムやデータがメモリに、格納されデバイス
（コンピュータ内部の装置や周辺機器 CPU やメモリ、ハードディスク、ビデオカードなどコンピュータを構成する各装置や、キーボードやマウス、プリンタ、ディスプレイなどの周辺機器）へアクセスできるようにする為に、カーネルは、マザーボードの入出力も管理する。

カーネルのする事は、アプリケーションプログラムの実行を許可して、ハードウエアを、抽象化して、アプリケーションプログラムをコンピュータのメモリに格納し、格納されたプログラムを、指定されたとおりに、実行させます。
カーネルは、システムの全メモリへのアクセスが可能です。アプリケーションプログラムなどは、規制があり、システムの全メモリへのアクセスが規制されています。また、カーネル内のモジュールやデバイスドライバが、使用するメモリの割り当てをする。

モジュール：機能単位、交換可能な構成部分という意味の英単語。システムへの接合部(インターフェース)が規格化・標準化されていて、容易に追加や削除ができ、ひとまとまりの機能を持った部品のこと。

デバイスドライバ：周辺機器を動作させるためのソフトウェア

## システムコール

プログラマーなどが、直接　システムコールを使って、カーネルを操作する事が出来ます。
われわれも、プログラムを組んで、pc につながっているプリンターや、モニタなどのハードウエアを
コントロールする事も出来ます。
　システムコール：オペレーティングシステム OS のカーネルの機能を呼び出すために使用される機構。

## シェル

カーネルとは、名前のとおり、核、または、中核などと言うように大事なもので、あまり直接触れることはありませんが、普段はシェルという仲介を通して、接します。
シェルは、人間とコンピュータの仲介です。

今まで、いろいろな、コマンドを実行してきましたが、コマンドを実行する時に、　% や　# などの
プロンプトが、表示されていました。
プロンプトが、表示されていることは、コマンドの入力を許可されているという事です。
プロンプトを表示して、ユーザの入力したコマンドをカーネルに通訳するのが、シェルです。

シェルには、（ｓｈ）、ｃｓｈ、　ｔｃｓｈ、ｂａｓｈ、ｚｓｈ　などのシェルがあります。

ここでは、ｃｓｈ　ｃシェルを、見ていきたいとおもいます。

# touch

ファイルのアクセス時刻と変更時刻を変える
書式

## touch　[-cm] [-r file] [-t [[CC]YY]MMDDhhmm[.SS]] file ...

touch は、file　で指定したファイルのアクセス時刻と変更時刻を、現在の時刻に変える。
ファイルが存在しないときは、デフォルトのパーミッションで，サイズ０のファイルが
できる。

　デフォルト：（default）利用者が何も操作や設定を行なわなかった時に使用される、まえもって組み込まれた設定値。「初期設定」「既定値」

　パーミッション：（permission）コンピュータのハードディスクなどに保存されているファイルやディ

レクトリに対するユーザのアクセス権のこと。

オプション

-c　ファイルが存在しなかった時、ファイルをつくらない。touch コマンドは、その時エラー扱いをしません。エラーメッセージは端末（モニター）に出力せずに、戻り値にも影響しない。

-m　ファイルの修正時刻を変える。最終更新日時のみを変更します。

-r　ファイルのアクセス時刻や修正時刻を設定する時に、現在時刻ではなく、
file(コマンド)のアクセス時刻／修正時刻をもちいる。
最終更新日時を file の日時にあわせる。

-t time　最終更新日時を time に変更する。
[-t [[CC]YY]MMDDhhmm[.SS]]で変更時刻を指定する。
CC　西暦の千と百の位（世紀）
YY　西暦の十と一の位　（YY を、指定して、CC を省略した時、YY が３９－９９の間なら、
1939 年から 1999 年が、指定され、それ以外は、２１世紀の年とみなされる。
MM　月　１月から１２月
DD　日　　１から３１日
mm　分　0 から 59
SS　秒　　0 から 59
CC と YY が指定されていないと、現在の年になります。
SS がしていされないと、値は0になる。

使用例

ファイルの最終更新日時を、2008 年 08 月/24 日　21 時 34 分に変更
% touch　–t　200808242134　file

file が存在しないとき、空の file を、新たに、作成します。
% touch file

# csh (c シェル)　1

## メタキャラクタ

メタキャラクタとは、シェルが解釈する特殊な文字のことです。
どのようなものがあるかを、みていきます。

「*」： アスタリスク
「*」は、0 文字以上の任意の文字列を表すメタキャラクタです。
例えば、　*.doc とあれば、 * 以降右の　.doc で終わる全てのファイル名に置き換えられます。
「*」は、0 文字の文字列にも該当するので、
例えば、　bag.* とあれば、　*以前の　bag ではじまる名前のファイルがあれば、それも含まれます。
「*」は、ファイルの途中に入れる事も可能です。

touch コマンドで、ファイル anko, ame, purin, banana, hotdog, oden , yakisoba などのおやつのファイルをつくって、* を、ls コマンドなどを使って　試してみましょう。

```
% touch anko ame purin banana hotdog oden yakisoba
% ls
ame             banana          oden            yakisoba
anko            hotdog          purin
%   ls *a
banana          yakisoba
% p*
purin
%ls *d*
hotdog          oden
%ls *do *
hotdog
%
```

ls *a は、a で終わるファイル名を表示します。
ls p* は、p で始まるファイル名を表示します。
ls *d* は、ファイル名の途中に d が入っているファイルを表示。
ls *do*は、ファイル名の途中に do が入っているファイルを表示。

任意：●その人の意思に任せる事。●そうするか否か、どれにするかが、勝手に選べる事●勝手に選ばせる事

「?」：クエスチョンマーク
「?」　は、任意の１文字を表すメタキャラクタ
「?」　は、「*」と違い、０文字の文字には、当てはまりません。

例えば　t?oufu.doc　（豆腐）と、指定したときに、toufu.doc は、当てはまりません。
?のところには、任意の１文字が入らなければいけません。

先ほど作ったファイルで試して見ます。

```
% ls
  ame           banana        oden          yakisoba
anko            hotdog        purin
% ls p?rin
purin
% ls *o?a
yakisoba
%ls b?anana
ls: No match
% ls a??
ame
% ls a???
anko
%
```

ls p?rin は、p ではじまり、？の任意の１文字にあたる文字の次に rin で終わるファイルが、purin だけです。

ls *o?a　は、* は任意の０文字以上の文字列に当てはまり、次の？は任意の１文字にあたり、最後は a で終わるファイルは、yakisoba に該当します。

最後に、ls b?anana は、?の所には、必ず１文字を入れなければならないのですが、入れると、該当するファイルが見つかりません。したがって、No match とマッチしないと、言われてしまいます。
ls a??　と　ls a??? を、比べてみてください。　ls a?? は、a で始まり任意の２文字で終わるファイルに該当し、　ls a???は、a で始まり任意の３文字で終わるファイルに該当します。

「?」は、任意に１文字に、該当するので、「*」と違い、「?」　の部分に該当する文字が、必ず無いとだめです。

「[]」：角括弧　（プラケット）
「[]」は、「[]」の中に文字の候補を指定するメタキャラクタです。
「[]」は、そのなかに、入れた文字の中のいずれかの１文字として、解釈されます。
例えば　[2 3 b]が、中に入ったとすると、2 か、3 か　b　の内のどれか、１文字に該当します。

「[]」の中にかく文字の候補は、「-」で、範囲を、決める事ができます。
例えば　[2-4]は、2, 3, 4 が、該当します。
　　[0-9]は、0, 1, 2, 3, 4, 5, 6, 7, 8, 9 が、該当します。
　　[a-d]は、a, b, c, d が、該当します。
　　[a-z]は、a,b,c,d,e,f,,,, ,,,q,r,z までが、該当します。

先程のファイルを、touchk コマンドを使って増やしてから、見ていきます。

```
% touch a b c d 1 2 3 4 z.jpg z.html z.text
% ls
1               a               banana          oden            z.jpg
2               ame             c               purin           z.text
3               anko            d               yakisoba
4               b               hotdog          z.html
% ls [4a]
4       a
% ls [a-z]
a       b       c       d
% ls [a-c]*
a       ame     anko    b       banana          c
%   ls   z.{html,jpg}
z.html          z.jpg
%
```

最後の　ls [a-c]*は、頭が a か、b か, c のファイルに該当します。

「{}」：波括弧　（ブレース）
「{}」は、「{}」の中に、文字列を「,」で区切って文字列を指定します。

ls z.{html,jpg}は、z.html と　z.jpg を表示する事です。

ここで、少し面白いものを、見つけました。
見てください。

｛｝のなかの「,」（カンマ）の後には、空白を入れない事に、注意してください。カンマの後に空白を入れると、エラー表示が出てきます。

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

これらの、結果を見てみると、カンマ　「,」で区切られた文字列１つを処理して、次の文字列を処理する。
そして、最後に結果を、合計して、アルファベット順に表示しているみたいです。

ls {a,a} と、波括弧のなかに、a,a と a を二つ入れると、｛a｝の結果が２つ表示されます。
a　　a

ls {a*,a*}　では波括弧のなかに、a*,a*と a*を二つ入れると、｛a*｝の結果が２つ表示されます。

a　　a　　ame　　ame　　anko　　anko

% ls {a*,b*,a*}は、波括弧のなかに a*と a*を２つ入れ、b*を１ついれました。b*は、a*と a*の間に挟んで入れてみます。
結果は、｛a*｝の結果が２つと、｛b*｝の結果１つの合計の結果が、アルファベット順に表示しているみたいです。

a　　a　　ame　　ame　　anko　　anko　　b　　banana

「^」：カレット

「^」は、該当する物以外のメタキャラクタを指定する。

「^」は、文字列の先頭に使うと、ファイル名を、否定して、指定をします。

例えば　^4ab は、4ab のファイル以外の事を示し、＾bg2*　ならば、先頭が bg2 以外全てのファイルを示します。

「^」を、文字列の先頭以外に使うと、一つの文字として扱われる

```
% ls *n
oden      purin
% ls ^*n
1                 a                 banana            yakisoba
2                 ame               c                 z.html
3                 anko              d                 z.jpg
4                 b                 hotdog            z.text
%ls {a*}
a         ame     ank
% ls {^a*}
1                 4                 c                 oden              z.html
2                 b                 d                 purin             z.jpeg
3                 banana            hotdog            yakisoba          z.text
% ls
1                 a                 banana            oden              z.jpg
2                 ame               c                 purin             z.text
3                 anko              d                 yakisoba
4                 b                 hotdog            z.html
% ls {^a*,a*}
1                 a                 banana            oden              z.jpg
2                 ame               c                 purin             z.text
3                 anko              d                 yakisoba
4                 b                 hotdog            z.html
% ls {^a*, a*,b*}
1                 a                 b                 d                 yakisoba
2                 ame               banana            hotdog            z.html
3                 anko              banana            oden              z.jpg
4                 b                 c                 purin             z.text
%
```

先ほどの ls *n は、n で終わるファイルを表示させます。
ls ^*n は、n で終わらないファイルを表示させます。

次は、
ls {a*}は、a で始まるファイルを表示させます。
ls {^a*}は、a で始まらないファイルを表示させます。

次は、
ls です。この ls だけの結果と、下記の
ls {^a*,a*}の結果を、見比べてください。同じです。

最後の
ls {^a*,a*,b*}は、先ほどの ls {^a*,a*}の結果と ls {b*}の結果を足したものです。

「~」：チルダ
「~」は、パスで使用する時に、ホームディレクトリを表すメタキャラクタです。
「~」は、自分のホームディレクトリとシェルは、解釈します。
例えば　cd ~ と　コマンドを打つと　自分のホームディレクトリに、移動します。
また、
「~」の後ろにユーザ名を入力すると、ユーザのホームディレクトリと解釈されます。

まずは、cd コマンドで、自分のホームディレクトリ移動して、pwd コマンドを使って、自分のホームディレクトリを確認します。ちなみに、わたしのホームディレクトリは　/usr/home/hiro です。
他のユーザのホームディレクトリへの移動は、cd ~ユーザ名　で、移動できます。ここでは、
/usr/home/user

```
% cd
% pwd
/usr/home/hiro
% cd /
% pwd
/
% cd ~
%pwd
/usr/home/hiro
% cd ~user
%pwd
/usr/home/user
%
```

書式

# echo[-n]文字列

引数の文字列を、標準出力で表示する。

echo　は、引数の間を、１つの空白文字で区切って、最後に改行文字を付けた文字列を
標準出力で表示する。

オプション
-n：文字列の最後に改行を表示しない。（改行文字の手前までを表示する。）

例

ohayou と標準出力
% echo ohayou
ohayou

ｎで終わるファイルを標準出力
% echo *n
mikan　　odenn

次に、echo コマンドで、「？」(クエスチョンマーク)を表示してみましょう。

```
% echo mikan tabetai
mikan tabetai
% echo ?
1 2 3 4 a b c d
%
```

echo コマンドでは、「？」(クエスチョンマーク)を表示する事ができずに、メタキャラクタの役割をして、任意の一文字のファイル　1　2　3　4　a　b　c　d　　を、表示してしまいます。

「？」(クエスチョンマーク)を表示するには、

「'」：シングルクォート
「'」と「'」で挟まれたメタキャラクタ文字は、メタキャラクタの解釈をシェルは行わない。

「"」：ダブルクォート
「"」と「"」で挟まれたメタキャラクタ文字は、メタキャラクタの解釈をシェルは行わない。

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

touch *では、ファイル* を作る事ができませんでした。
ファイル*をつくるには、"
　touch '?' と、「?」を「'」と「'」の２つで、挟まなければなりません。
すると、ls コマンドで見てみると、ファイル?　が、出来ています。
また、
「\」：バックスラッシュ
「\」も「\」のすぐ後のメタキャラクタの特殊性を、無効にする働きがあります。

## ヒアドキュメント

“<<” は、ファイルからのデータ入力ではなく、指定したデータをコマンドの入力データとして、使用します。

使い方は、 コマンドの後ろに“<<”を入力して、 “<<”の後ろに自分で指定した、単語を入力して、実行します。

すると、？が表示されます。？が表示されましたら、自分で、データを入力して Enter キーを押します。

最後に、始めに使った同じ単語を入力すると、終了です。

| | |
|---|---|
| コマンド　<<　単語 | ・・・・・・単語で始まる。 |
| ？ | ・・・入力データ。 |
| ？ | ・・・入力データ |
| ？ | |
| ？ | |
| ？ | |
| 単語 | ・・・・・単語で終わる。 |

例えば

```
cat << osirase
? kyouwa
? amedesu
? osirase
kyouwa
amedesu
%
```

「;」：セミコロン
「;」は、コマンドを連続させる働きがあります。

例えば

cd .. ; pwd ; ls

まず、下記のような、作業を

```
% pwd
/usr/home/user
% mkdir dirA
% cd dirA
% pwd
/usr/home/user/dirA
% touch memo
%ls
memo
%
```

下記のようにに、短縮できます。

```
% pwd
/usr/home/user
% mkdir dirA ; cd dirA ; pwd
/user/home/user/dirA
% touch memo ; ls
memo
%
```

両方とも、現在のカレントディレクトリを確認して、ディレクトリ dirA をつくり、cd コマンドで、dirA に移動して touch コマンドで、ファイル memo をつくり、ls コマンドで、確認をする。

「`」：バッククォート
「`」と「`」でコマンドを挟むと、コマンドの結果を入力として使用できる。

例えば

echo ` ls`

echo `date`

```
% echo ` ls`
1 2 3 4 ? a ame anko b banana c d hotdog oden purin yakisoba z.html z.jpg z.text
% echo `cal`
September 2008 Su Mo Tu We Fr Sa 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
% echo `date`
Web Sep 10 16:819 UTC 2008
%
```

## which　コマンド

コマンドのパスの検索結果を表示する。

which ls のコマンドの結果（パス（path））をつかい。ls コマンドを使って、ファイル ls の
更新日時や、ファイルのオーナ、保護モードなどを見る。

```
% which ls
/bin/ls
% ls -l ` which ls`
-r-xr-xr-x　1　hiro　group　23564 Jan 16 2008　/bin/ls
%
```