



もう少し進んだシェルコマンド

知っていると便利なコマンドをもう少し。



シェルの種類

- シェルとは何か覚えていますか？
- ボーンシェル (Bourne shell, sh)
 - AT&T の Steve Bourne が作ったUnix最初のシェル
- コーンシェル (Korn shell, ksh)
 - AT&T の David Korn が作ったUnix初期のシェル
- C-shell (csh)
 - Bill Joy が Berkeley Unix のために作ったシェル。
- Turbo C-shell (tcsh)
 - C-shell のユーザーインターフェースが強化されてる。
- Bourne again shell (bash)
 - Linux の標準シェル。Bourne shell との互換性も高い

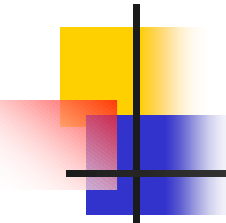


shell の使い方

- Shell はプロンプトを表示する。
- コマンドをタイプする。
- リターンキーを押す。
- シェルはタイプされたコマンドを解釈し、正しいプログラム見つけてそれを実行するようにカーネルにリクエストする。
- カーネルがリクエストされたプログラムを実行させ、その結果をシェルに返す。
- シェルは実行結果を表示する。その後、再びプロンプトを表示する。

標準入出力とエラー

- 標準入力 (standard input)
 - stdin
 - プログラムが通常の入力 (input) を探しにいくところ。
 - キーボード。
- 標準出力 (standard output)
 - stdout
 - プログラムが通常の出力を表示するところ。
 - スクリーン (ターミナル画面。)
- 標準エラー出力 (standard error)
 - stderr
 - プログラムがエラーメッセージを表示するのに使うところ。
 - これも、スクリーン。



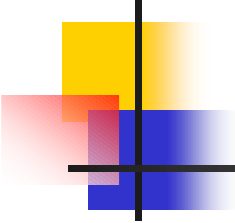
リダイレクト <, >, >>

- <

- 標準入力をリダイレクトする。

- [コマンド] < [ファイル名]

- コマンドは指定したファイルを開いて、そのファイルの内容を入力ソースとして利用する。



リダイレクト <, >, >>

- >

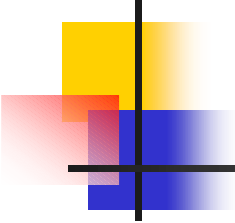
- 標準出力のリダイレクト

- [コマンド] > [ファイル名]

- コマンドの出力結果が指定したファイルへと送られる。

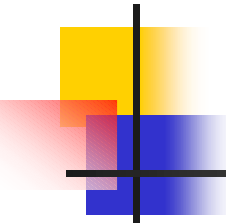
- 出力先のファイルは新たに作られるか、上書きされる。

- `cat june july august > 2007_summer`



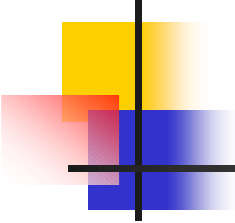
リダイレクト <, >, >>

- >>
- 標準出力をリダイレクトする。
 - [コマンド] >> [ファイル名]
- コマンドの出力結果が指定したファイルに書き込まれる。
- ファイルが既に存在する場合には、出力結果をファイルの最後に付け足す。



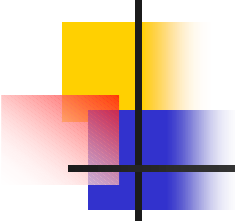
幾つかのコマンドをまとめて実行する(グループピング)

- 一つ一つのコマンドを順番に実行するのが面倒なこともある。
- コマンドをセミコロン";"で区切って並べてやることで、コマンドをまとめて実行することができる。
 - `pwd; cal 4 2007; date`



| (パイプ、pipe)

- リダイレクトとグルーピングに似ている。
- コマンドをリンクするのに使う。
 - [コマンド] | [コマンド] etc.
- 最初のコマンドの出力が2番目のコマンドの入力に送られ、その出力が... となる。
 - who | more



ワイルドカード (Wild-card)

- コマンドを全部タイプするのがめんどくさい時もあるさ。
- そんな時、3種類のワイルドカード文字を使う:
 - アスタリスク (Asterisk “*”): 全ての文字列 (スペース (空白) も含む) とマッチする。
 - はてなマーク (Question mark “?”): 1文字とマッチする。
 - 四角括弧 (square bracket “[]”): 括弧の中の全ての文字とマッチする。
- 特殊文字を引用する。



WC

- 単語を数える。(ワードカウント、word count)
- ファイル中の単語数を表示する。
 - `wc [-c | w] [ファイル名]`
- 出力は、行数、単語数、及び文字数。
- フラッグを付けて出力の内容を変更できる。



sort 並べ替え(ソート)

- ファイルの内容を並べ替える。
 - `sort [-b f n r] [ファイル名]`
- ファイルの内容を取り出し、それを並べ替えた順序で表示する。
- フラッグ(Flags):
 - `-b`: 空白を無視する
 - `-f`: 大文字と小文字を区別しないで扱う。
 - `-n`: 数値でソート
 - `-r`: 逆順序で出力する



ジョブの制御

(ジョブコントロール、Job control)

- Unix では、ジョブ (job) とかプロセス (process) と呼ばれるものが、走っている (実行されている)。
- 全てのコマンドやプログラムは、ユーザーによって実行された、別々のジョブ / プロセス (job/process) である。
- ジョブは普通フォアグラウンドで走っている。もちろん、バックグラウンドで走らせることもできる。
- ジョブを流したユーザーは、そのジョブを停止 (kill) させることもできる。

ジョブの制御

(ジョブコントロール、Job control)

- ctrl-c: コマンド/ジョブをキャンセルする。
- ctrl-z: コマンド/ジョブを一時停止させる。
- jobs
 - 自分が現在実行中のジョブ(プログラム)のリストを表示させる。



bg

- ジョブをバックグラウンドで走らせるようにする。
- そのためには、まず、ctrl-z をタイプしてジョブを一時停止させる。
- 次に、bg をタイプする。そうすると、そのジョブはバックグラウンドで走るようになる。
- jobs コマンドでバックグラウンドで走っているかを確認できる。
- コマンドを実行する際に、最後に&を付けることで、ジョブを初めからバックグラウンドで実行するようにすることもできる。



fg

- バックグラウンドで走っているジョブをフォアグラウンドにする。
- まず、jobs コマンドで、自分が実行中のジョブを確認する。
- fg %[ジョブの番号] とタイプすると、そのジョブがフォアグラウンドになる。



kill

- 自分が走らせているジョブをキャンセルする。
- jobs コマンドで自分が走らせているプログラムを確認する。
- kill %[ジョブ番号]とタイプする。
- 他のやり方
- ps -u [ユーザー名] でプロセスID番号を確認。
- kill -9 [プロセスID番号] とする。