

システム管理論

-- コマンドインタープリター --

栗野 俊一

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く
禁じます

2026/05/26 システム管理

私語は慎むように !!

□ 席は自由です

- できるだけ前に詰めよう
- コロナ対策のために、ソーシャルディスタンスをたもとう

□ 色々なお知らせについて

- 栗野の Web Page に注意する事

<http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino>

- google で「kurino」で検索

前回の復習

システム管理論

前回の復習

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

前回の復習

□ 前回の復習

○ 講義: ディスクのパーティション

- ▶ ディスクを複数のパーティションに分ることができる
- ▶ 一つのディスクに複数のファイルシステムを作る事ができる

○ 実習: スクリプトの作成

今回の概要

システム管理論

今回の概要

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

本日(2026/05/26)の予定

□ 本日(2026/05/26)の予定

○ 講義：コマンドインタプリター

▶ Shellをコマンドインタプリタとして捉える

▶ 標準入出力, プロセス, Shell変数, 環境変数, 正規表現

○ 実習

▶ Bash 上の操作

今日(2026/05/26)の目標

□ 今日(2026/05/26)の目標

○ 講義

- ▶ コマンドインタプリター

○ 実習

- ▶ Bash の操作に馴れる
- ▶ Shell Script の作成

本日の課題 (2026/05/26)

- 前回 (2026/05/19) の課題
 - 講義中に作成した Script の提出
- 今週 (2026/05/26) の課題
 - 講義中に作成した Script の提出

コマンドインタープリター

システム管理論

コマンドインタープリター

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

コマンドインタプリター

□ コマンドインタプリター

- コマンドを解釈(インタプリター)して、実行するソフト

 - ▶ shell は コマンドインタプリター の一種

- ファイルから命令列(プログラム)を読み込んで実行可能

 - ▶ shell script

□ コマンドインタプリター としての bash

- 状態 (変数とその値) をもち、それに合わせて動作を変更する

- unix(Linux) の様々なリソース(資源)を参照、操作可能

プロセス

□ プロセス

- **unix** おける、実行の単位 (実行中のプログラムと考えてよい)

- ▶ 実際は、プロセスの属性として、「プログラム」がある

□ プロセスの作成

- **fork** システムコール

- ▶ 子プロセスを作る (属性は引き継がれる) / 親プロセスが得られる

- **exec** システムコール

- ▶ プロセスで利用するプログラム変更する

- **system** : 新しいプログラムの実行

- ▶ fork して exec する

プロセス管理

□ プロセス管理

○ プロセスの属性

- ▶ PID : プロセス ID
- ▶ TTY : プロセスに割り当てられている TTY [I/O (標準入出力)]
- ▶ COMMAND : プロセスに割り当てられているプログラム (コマンド)
- ▶ USER : プロセスの持主

□ プロセスの状態確認

○ ps コマンド : プロセスの状態を把握する

○ /proc ファイルシステム : プロセスの情報がファイルの形で参照可能

- ▶ /proc は、最近の unix (Linux 等) の機能

○ fore grand / back grand / suspend

□ プロセスの操作

○ プロセス作成/中断(サスペンド[Ctrl-Z])/終了[Ctrl-C/kill]

- ▶ jobs / '&' / '|' / bg / fg

□ シグナル

○ プロセス間で、信号(シグナル)を送る仕組み

- ▶ シグナル例 : INT(割り込み)/KILL(強制終了)/HUP(端末の中断)

○ kill コマンド : シグナルを送るコマンド

標準入出力

□ 標準入出力

○ プロセスの属性の一つで、入出力(I/O)を示す

▶ 標準入力 (0/stdin) : 標準の入力 (tty : キーボード)

▶ 標準出力 (1/stdout) : 標準の出力 (tty : ディスプレイ[バッファード])

▶ エラー出力 (2/stderr) : エラーの出力 (tty : ディスプレイ[アンバッファード])

□ リダイレクション(標準入出力の切替)

○ プロセスの標準入出力が変更できる

▶ '<' 入力切替

▶ '>' 標準出力切替 / '>>' 切替と同時に追加

▶ '2>' エラー出力 / '&>' エラーを標準へ

○ パイプ ('|')

▶ 複数のプログラムの入出力を継ぐ

▶ grep : 文字列のパターンを探す

▶ sed : 文字列を操作する

▶ more/less : 出力の表示操作

変数

□ 変数操作

- '=' で、値を設定 / '\$' で値を参照 / set で一覧

□ Shell で扱える変数

- Shell変数 : Shell (だけ) が (直接) 管理する変数

- ▶ Shell (や、そのスクリプト) 内でしか意味がない
- ▶ 自分で自由に追加、役割を決める事ができる
- ▶ Shell 固有の役割を持つ変数もある (例 : \$PS1, \$?)

- 環境変数 : プロセスに割り当てられている変数

- ▶ 子プロセスに引き継がれる (export コマンド)
- ▶ export で設定 / env コマンドで確認

正規表現 (RE: Regular Expression)

□ 正規表現 (RE: Regular Expression) とは

- 文字列のパターン(集合)を簡単に表現したもの (cf. ワイルドカード)

- ▶ 一つの文字列(正規表現)で、複数の文字列(文字列の集合)が表現できる

- ▶ 例: `file-*.txt` => "file-" で始まり、".txt" で終わる文字列 (`file-1.txt`, `file-abc.txt`, `file-file-file.txt.txt`, ...)

□ 正規表現の基本

- 普通の文字一文字 (A) : その文字そのものを表す ({ "A" })

- ▶ メタ文字 (特別な意味のある文字 [. ^ \$ [] * + ? | ()]) は、\ でエスケープ

- ▶ `\[` : "[" 自身 / `\\` : "\" 自身

- 文字の範囲指定 ([A-Z]) : 文字の集合を表す ({ "A", "B", ..., "Z" })

- ▶ . は、「任意の文字」の意味になる

- ▶ 先頭の ^ は否定になる (`^[A-Z]` は [A-Z] 以外)

- 正規表現の並び (AB, [A-C][1-3]) : 集合の直積になる ({ "AB" }, { "A1", "A2", ..., "C3" })

- 正規表現の選択 (AB|[1-3]) : 和集合になる ({ "AB", "1", "2", "3" })

- 正規表現の繰返し (A*) : 0 回以上の繰返し ({ "", "A", "AA", .. })

□ bash の正規表現

- 正規表現には、方言 (コマンド毎に異なる) がある : 機能は同じだが表現が違う

- ▶ 例 : `bash` の `*` / `.` は、`grep` の `*` / `\.` と同じ意味

おしまい

システム管理論

おしまい

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます