重要な環境変数（私見）

- PWD
  - カレントディレクトリの絶対パス
  - カレントディレクトリに合わせて動作を変えるスクリプトを作るときに使う

- SHELL
  - 現在ユーザが使っているシェルのパス
  - /bin/bash や /usr/bin/tcsh など
  - シェル特有の機能を使うときに確認する

シェルスクリプトの特徴(1)

- 自動化！自動化！
  - ➢複数のコマンドを1つにまとめれる
  - ➢組み合わせると大体なんでもできる
    - ✓データのバックアップや日付管理
    - ✓ログの調査，加工，管理
    - ✓サーバの監視
    - ✓プログラムの自動テスト
    - ✓(システムへの攻撃)
  - ➢**日々の既定作業を減らせる**

シェルスクリプトの特徴(2)

- コマンドをそのまま使える
  - コマンドは他者が作ったプログラムである
  - 1つ1つが完成した機能を持つ
  - コマンドは実行するだけで結果を得られる
  - それらを組み合わせるだけ

ファイルを読み込むには？

```
FILE *fp=fopen(“piyo.txt”,”r”);
while(fscanf(fp,…,data);…
```

C

```
open(fh,“<”,“piyo.txt”)
my $data= do{ local $/; …
```

Perl

```
with open(“piyo.txt”) as f:
    data = f.read()
```

これだけ！

```
data=`cat sample.txt`
```

シェルスクリプトの特徴(3)

- 移植性が高い
  - コマンドが動けばどこでも動く
  - シェルがインタプリタ(逐次解釈)する
  - シェルと使うコマンドを導入したら大丈夫！
  - POSIXに準拠したコマンドを使うことが重要

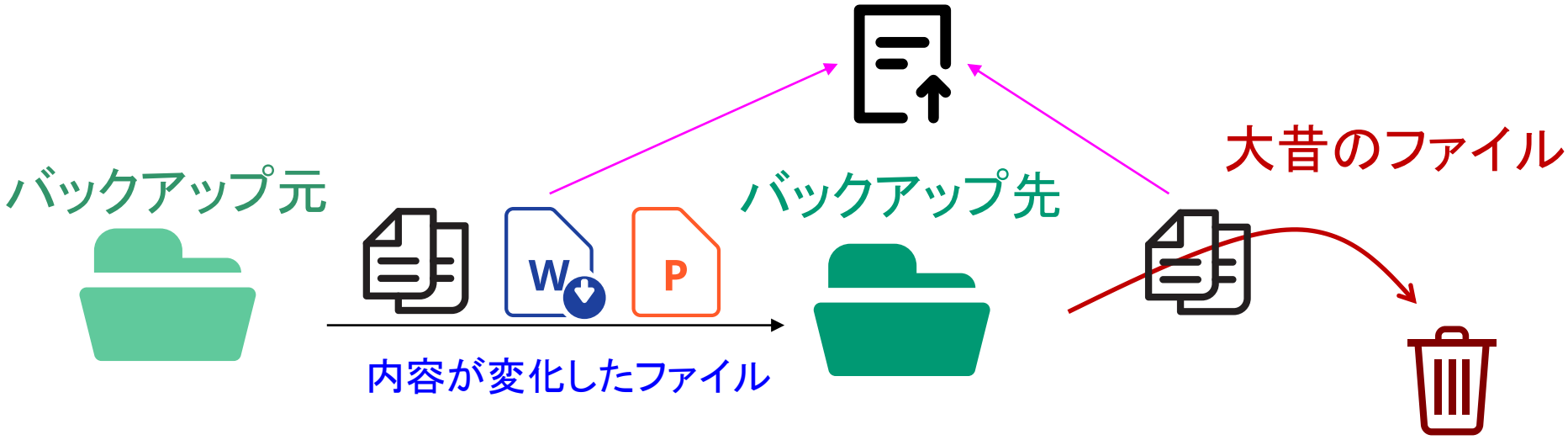
- POSIX
  - UNIX系OSが備えるべき標準規格
  - Linuxは概ね準拠している
  - 大体shなら大丈夫

シェルスクリプトの特徴(4)

- 遅い
  - インタプリタの宿命
  - 将来のコンピュータの性能では(多分)問題ない
  - アルゴリズムを捻る
    - ✓キャッシュを作っておく
    - ✓ファイルへの無駄なアクセスをへらす
  - 既存のコマンドを使う
    - ✓for loopは遅い
    - ✓sedやawkはバイナリレベルで高速にloopしてくれる

時間があれば作ってみよう！

- バックアップ元とバックアップ先を比較する
  - ➢内容に変化があるファイルを求める
- 変化があるファイルのみをバックアップする
- 一定期間以上前のファイルは削除する
- 操作したファイルをログに出力する



シェルスクリプトのデバッグ

shのオプション を使う

```
sh -n -v -x -u -e スクリプトファイル
```

コマンドのように実行するなら `#!/bin/sh -n` など

- n: シェルスクリプトの文法をチェックする.
- v: これから実行するコマンドを表示する. 変数は展開されない.
- x: 実行したコマンドを出力する. 変数なども展開される.
- u: 変数が未定義のときエラーで終了する
- e: コマンドの終了ステータスが0以外の時, スクリプトを強制終了する

シェルスクリプト作成のコツ

- 最終的にどのような出力がいるか考える
  - 標準出力やファイル出力が含まれる
- 出力は何種類あるか考える
- どのコマンドの結果で得られるか考える
  - コマンドは何個使っても良い
- コマンドの結果のどこが欲しいか考える
- 不要な箇所を削る
- 最後に組み合わせる

expect

- 対話プログラムと“対話”できる
  - プログラムからの質問に応じて指示できる
  - 制御構造が使える
- 対話の自動化の一例
  - passwdやsudoのパスワード入力
  - ssh先とも対話できる
    - ✓ログイン，先でコマンド実行，結果受け取り
  - ftpサーバにログインしてファイルの取得

ginにはインストールされていないので，
仮想マシンにインストールして使うこと