

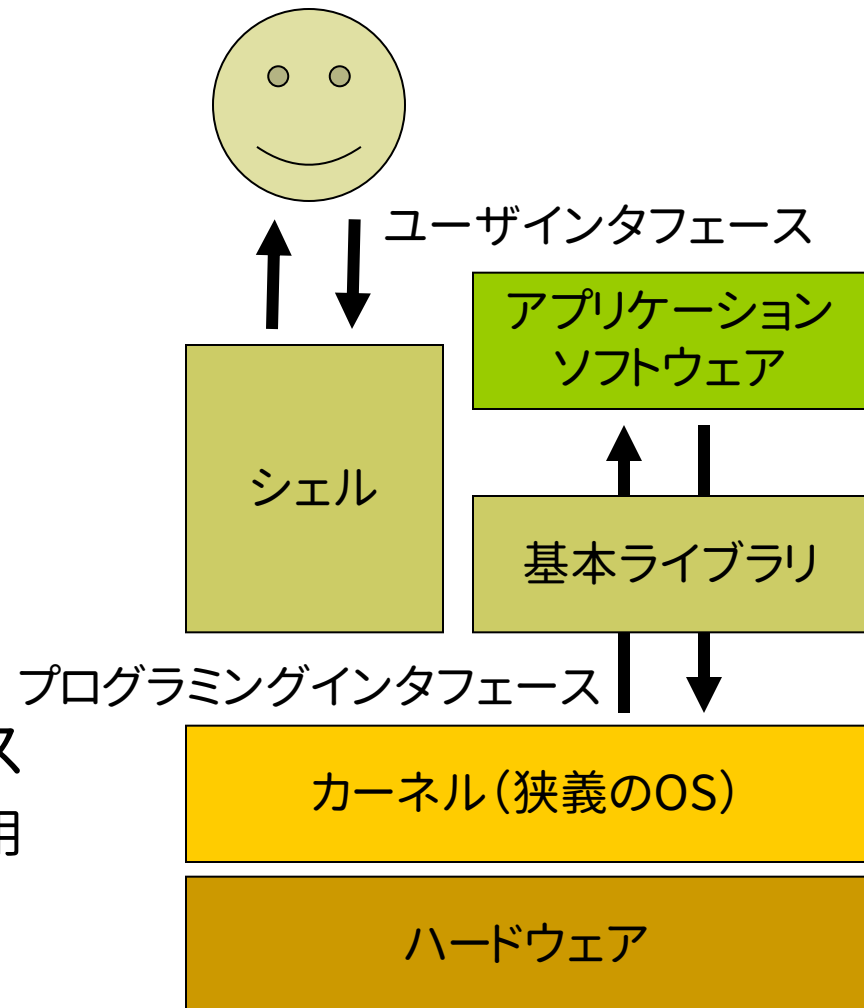
Operating Systems



第2回 OSのインタフェース
塩澤 秀和

OSのインタフェース

- Interface=「界面」
 - 2つのもの(層)の境界
 - それらの間で情報をやり取りする決まりごとや接続方法
- ユーザインタフェース
 - ユーザがコンピュータを使うときの操作方法
- プログラミングインタフェース
 - ソフトウェアがOSの機能を利用するときの方法
 - 関数ライブラリ, ファイル形式



プログラミングインタフェース

□ API/システムコール

- 各ソフトウェアがOSの機能を利用する(“呼び出す”)ときの方法
- たとえばC言語なら,関数(メソッド)や構造体(クラス)の取り決め
- API = Application Program Interface (OS以外にも使う用語)

□ 実行ファイル形式

- 実行可能なソフトウェアの内部形式
- コンパイラは,ソースコードからOSに対応した実行ファイルを作る
- 例) WindowsならEXE (PE)形式,UNIX/LinuxならELF形式など…

□ その他の実行環境

- プログラムを実行するときに,OSが提供するものすべて
- メモリの割り当て方法の決まりなど…

API/システムコール

- API/システムコールで提供される機能
 - プログラム(プロセス)の起動や終了
 - ファイルのオープンやクローズ,データの読み書き
 - フォルダ(ディレクトリ)の作成やファイル情報の管理
 - 画面表示,キーボード入力の取得,音の再生
 - プロセス間通信やネットワーク通信
 - ウィンドウシステムの制御
 - セキュリティ(ユーザ認証,データ暗号化など)

- API/システムコールの例
 - Windows (Win32) API, .NET Framework API, UWP API
 - UNIX/POSIXシステムコール, macOS Cocoa/Core○○
 - Java API, Android API, etc...

APIを調べてみよう

- Windows (Win32) API
 - Windowsの伝統的なAPI
 - <https://docs.microsoft.com/en-us/windows/win32/>

- UNIX / Linux システムコール
 - UNIX系OSの機能は, POSIXで標準規格化
 - https://linuxjm.osdn.jp/html/LDP_man-pages/man2/syscalls.2.html

- μ ITRON API
 - 日本で作られた組み込みOSの規格
 - [uITRON4.0.pdf](#) - HOSのサンプルを見てみよう

ユーザインタフェース

- グラフィカルユーザインタフェース (GUI)
 - ウィンドウシステムによるインタフェースなど
 - WIMP: Window + Icon + Menu + Pointer
 - 情報を表す画面上の絵や図を, マウスやタッチなどで操作する
 - 現実世界をモデルにした「直感的」な初心者向けのインタフェース
 - ソフトウェアの作成には手間がかかる
 - PC (macOS, Windows), スマホ, カーナビ等で採用され, 現在主流

- コマンド言語インタフェース
 - 文字による対話
 - 文法に従ったコマンド (命令) を, キーボードから文字で打ち込む
 - プログラミング言語に近いプロ向けのインタフェース
 - UNIXのシェル: sh, csh, ksh, bash...
 - Windowsの「コマンドプロンプト」(cmd.exe) ⇒ 起動してみよう

シェル

□ シェルの役割

- オペレーティングシステムの“殻”(shell) ⇔ kernel(核)
- OSへのユーザインタフェースを提供する
- ユーザの認証,プログラムの起動,ファイルの操作など…

□ コマンドシェル

- 文字のコマンド言語によるCUI(文字だけのユーザインタフェース)
- 旧来の大型機,UNIX,サーバーの操作
- Windows コマンドプロンプト / UNIX 仮想端末 (terminal)

□ グラフィカルシェル

- ウィンドウシステムによるGUI(グラフィカルユーザインタフェース)
- PC(個人用コンピュータ),スマートフォン,情報家電
- Windows Explorer / macOS Finder

【実習】Windowsのコマンドシェル

- Windowsのコマンドプロンプト
 - 田メニュー → [Windows システムツール] → [コマンドプロンプト]

- プロンプト (prompt)
 - 文字だけの黒い画面に入力を求める合図(プロンプト)が表示される
 - これに続けてキーボードから命令(コマンド)を入力できる
 - デフォルトではカレントディレクトリ(操作対象のフォルダ)を表示
 - 例) C:¥Users¥ユーザ名>

- コマンドの例
 - 下記のそれぞれのコマンドを入力し [Enter]キーを押してみよう
 - dir, cls, cd, tree
 - 内部コマンドの一覧は,「help | more」で見ることができる

シェルの文法

□ 基本文法

- コマンド名 引数1 引数2 引数3 ...
- コマンド名や引数にスペースが含まれている場合は""で囲む

□ 使用できるコマンド名

- シェルに内蔵されているコマンド (dir, cdなど)
- 記憶装置 (ストレージ) にあるプログラムファイル (*.exe) なら何でも
- 例) calc [ENTER] と入力してみよ

□ オプション引数

- コマンドの追加機能を指定する引数の一種
- 通例, UNIXでは「-」, Windowsでは「/」(または「-」) で始まる文字列
- 例) dir /?

コマンドの実行

- コマンドを実行してみて、動作から意味を推測してみよう
 - dir
 - dir ¥
 - dir "¥Program Files" /p
 - taskmgr
 - start C:¥

- GUI(エクスプローラ)と比較してみよう

- ファイル操作コマンド
 - copy ファイル名1 ファイル名2 # ファイルのコピー
 - ren 旧ファイル名 新ファイル名 # ファイル名の変更(rename)
 - move ファイル名 移動先名 # ファイルの移動

ワイルドカード

□ ワイルドカード

- 「どんな文字にも当てはまる文字」のことをいう
 - 元の意味は,トランプゲームの万能カード(たいていはジョーカーの役割)
 - (注意)シェルのワイルドカードは、「正規表現」とは別のもの
- シェルのコマンド入力で引数のファイル名を指定するときに見える

□ ?

- 任意の1文字に適合する
- 例) `dir a?.c` # `a1.c`, `ag.c` などに適合する

□ *

- 何にでも(任意の個数の任意の文字)に適合する
- 例) `dir *.txt` # `readme.txt`, `xyz.txt` などに適合する

リダイレクト

□ リダイレクト

- コマンドに対する入出力を, コンソール(通常キーボードや画面)から, ユーザが指定したファイルの内容に変えられる機能

□ 出力リダイレクト

- プログラムの出力結果をファイルに保存するのに用いる
- > ファイル名: コマンドの(画面)出力をファイルに保存する
- >> ファイル名: 上と同様だが, ファイルの内容を消さずに追記する
- 例) `dir ¥Windows > win.txt`

□ 入力リダイレクト

- < ファイル名: ファイルの内容をコマンドへのキーボード入力とする
- 例) `more < win.txt` # moreは1ページずつの表示

パイプライン

- パイプライン(パイプ)
 - コマンドの出力を,別のコマンドに入力する機能
 - 2つのコマンドを起動して,それらの入出力をまるでパイプ(管)で連結したように,前のコマンドの結果を後ろのコマンドへ流し込む

- 形式
 - コマンド名1 引数列 | コマンド名2 引数列
 - 例) `dir ¥Windows | more`
 - 例) `dir ¥Windows | find "dll" # findは文字列検索`

- フィルタ
 - 入力をパイプで受け取るように設計された種類のコマンド
 - `more, sort, (Windowsの) find, (UNIXの) grep` など

シェルスクリプト

□ コマンド処理の自動化

- コマンドによる一連の手順をテキストファイルに保存しておく…
- 一種のプログラムとして実行できる ⇒ よく使う手順を自動化できる
- Windowsの場合：拡張子を「.cmd」か「.bat」にする
- UNIX系の場合：ファイルをchmodコマンドで実行可能に設定する

□ Windowsの例

- 下記の内容で「now.cmd」という名前のテキストファイルを作成

```
@echo off
echo 現在の日付と時刻
date /T
time /T
```

- プロンプトで、「now」と入力すると実行される

環境変数

□ 環境変数

- ユーザ向けの設定を保存しておけるOSが用意している“変数”
- Windowsでは, [⚙️ 設定]→検索欄に「環境変数」と入力して検索
- 参照方法: %環境変数名% (UNIX/Linuxでは \$環境変数名)

□ 環境変数「PATH」(実行パス)

- コマンドライン入力外部コマンド(プログラム)を探す場所
- WindowsでPATHの値を表示: `echo %PATH%`

□ ディレクトリ(フォルダ)とパス(Path)

- ファイルの位置を示す(日本では\の代わりに¥を使うことが多い)
- Windowsの例: `C:¥Users¥taro¥Documents¥文書1.doc`
- UNIX/Linuxの例: `/usr/local/bin/bash`

【参考】Windows用のツール

- Windows Terminal
 - Windowsの新しいコマンドラインツール
 - <https://docs.microsoft.com/ja-jp/windows/terminal/>
- Windows PowerShell
 - コマンドプロンプトよりも強力なコマンドシェル
 - <https://docs.microsoft.com/ja-jp/powershell/>
- Windows Subsystem for Linux
 - Windows上でLinux (UNIX)システムを動作させる
 - <https://docs.microsoft.com/ja-jp/windows/wsl/>
- Sysinternals
 - Windowsのシステム状態を調査するツール集
 - <http://technet.microsoft.com/en-us/sysinternals/>
- API Monitor
 - ソフトウェアからのWindows API呼び出しを追跡表示する
 - <http://www.rohitab.com/apimonitor>

演習課題

- 課題 2a コマンドプロンプトの利用
 - 今回学んだWindowsのコマンドプロンプトを利用してみよう。

- 問題
 - ① 以下のコマンドの実行例を示し、簡単に機能を説明せよ。
del, mkdir, rmdir, ipconfig, exit
 - ② 「cd ..」「cd ..¥..」「cd ¥」が、それぞれどういう意味か説明せよ。
 - ③ ワイルドカードを用いて、画像ファイル(拡張子jpg,gif,png)だけを、あるフォルダから別のフォルダにコピーする実行例を示せ。

- 提出
 - 提出用Wordファイルの「解答」または「結果」の欄(どちらでもよい)に、
 - ①～③の実行例と説明のテキストをコピー&ペーストする。
 - 「OS02_学籍番号_氏名.pdf」という名前のPDFファイルで提出する。