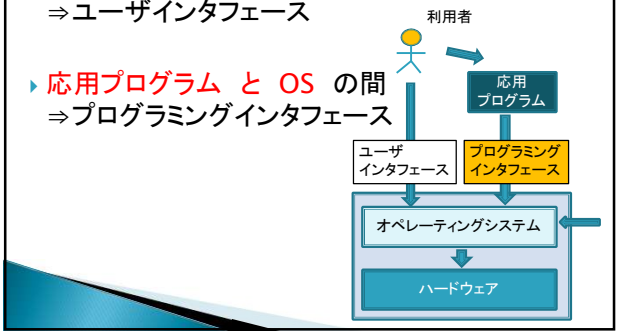
第3章 オペレーティングシステムのプログラミングインタフェース

花田 英輔
（このPowerPointは渡辺名誉教授作成のものを花田が一部改編した）

1

OSのインタフェース(教科書3.1)

- ユーザ と OS の間
  ⇒ユーザインタフェース
- 応用プログラム と OS の間
  ⇒プログラミングインタフェース

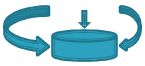


2

入出力プログラム例

```
#include <stdio.h>
#include <fcntl.h>
int main(){
  int fp;
  char c[20];
  fp = open("./test.txt", O_RDONLY);
  read(fp, c, 20);
  printf("%s", c);
  return 0;
}
```

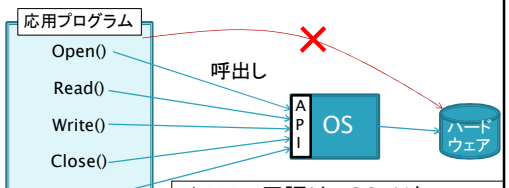
- ハードウェア関連の処理はOSに依頼してやってもらう
- それぞれのアプリが勝手に行うと混乱する



3

プログラミングインタフェースの目的

- OSの機能をプログラムから利用
- API（Application Program Interface）とも呼ぶ
  - 応用プログラム（Application Program）向けの受付窓口



✓APIの用語は、OS以外のソフトウェアモジュールについても用いられる

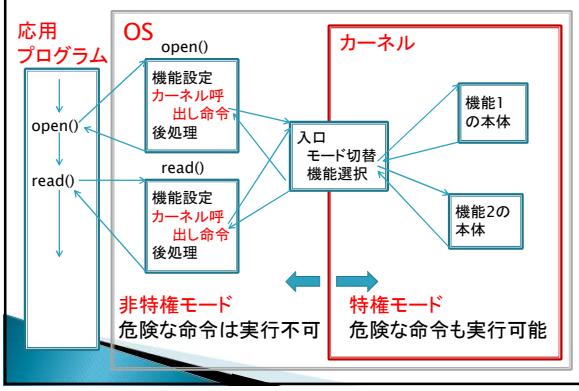
4

プログラミングインタフェースの提供 (教科書3.2)

- 使い方
  - 普通の関数呼び出しで利用
- 実現方法
  - OS重要部分（カーネル）は、応用プログラムと分離
  - 関数内でカーネルの呼出し＝カーネル呼出し命令
  - 詳細は次章
    1. カーネル呼出し命令(ハードウェア命令)を発行
    2. 割込み発生：応用プログラムを中断して割込み処理
    3. 特権モードへ移行
    4. OS本体内の処理ルーチンへジャンプ

5

プログラミングインタフェースの提供



6

APIルーチン

- UNIX:
  - システムコール関数＝Cの関数
- 汎用機OS:
  - OSマクロ
    ＝アセンブリ言語のマクロ命令
- Windows:
  - Win32API、Win16API、Win64API他、C++等から利用
  - GUI、マルチメディア等の機能も含む
- MacOS:
  - Cocoa（MacOS X）、Objective Cから利用
  - Carbon(Mac OS 9 以前)、C等から利用
  - その他

```
#include<windows.h>

int WINAPI WinMain(
    HINSTANCE hInstance ,
    HINSTANCE hPrevInstance ,
    PSTR lpCmdLine ,
    int nCmdShow ) {

    MessageBox(NULL , TEXT("Hello!") ,
        TEXT("MessageBoxTest") , MB_OK);

    return 0;
}
```

Win32プログラム例

7

C言語標準ライブラリとOS機能
(教科書3.3)

- UNIXとC言語は密接な関係
- UNIX
  - 当初はアセンブリ言語、直ぐに高水準言語で書きなおし
  - その時にOS記述用の高水準言語として考案＝C言語
- C文法中には入出力機能無し
  - 標準ライブラリ中に入出力を含むOS呼出しの関数群
- Cの標準化（ANSI）時、標準ライブラリも共に標準化
  - UNIX以外のOSの機能も同じインタフェースで可能

8

ANSI/ISO C言語規格 標準ライブラリ

- 入出力
  - fopen(), fclose(), remove(), rename(), printf(), scanf(), getchar(), fgetc(), fgets(), putchar(), fputc(), fread(), fwrite(), fseek(),
- ユーティリティ
  - malloc(), free(), exit(), abort(),
- シグナル
  - signal(), raise(),
- 日付と時間
  - clock(),time(),

元々はUNIXの機能、標準化に伴い他のOSでも同様に動く

9

UNIXのシステムインタフェース

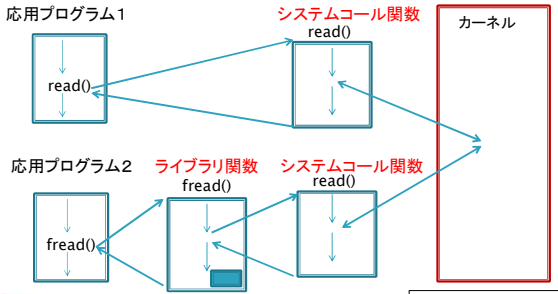
＝プログラミングインタフェース

- システムコール関数
  - ライブラリ関数より低レベル、カーネル呼出しを含む
- C標準ライブラリ内の関数
  - 内部でシステムコール関数を利用（利用しないものも）
- システムコール関数の例
  - open(), read(), write(), lseek(), ioctl(), close(), mkdir(), rmdir(), pipe(), chmod(), fork(), exec(), exit(), wait(), brk(),,

UNIX実装
他のOSで動くとは限らない

10

システムコール関数とライブラリ関数



ライブラリ関数はバッファ利用
プログラム中でfreadとreadを併用してはダメ

バッファ:
効率良く処理するため
データを溜めておく領域

11

システムコール関数とライブラリ関数

```
#include <stdio.h>
#include <fcntl.h>
int main(){
  int fp;
  char buff[20];
  fp = open("./test.txt",
                O_RDONLY);
  read(fp, buff, 20);
  printf("%s", buff);
  close(fp);
  return 0;
}
```

システムコール関数使用

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  FILE* infile;
  char buff[20];
  infile=fopen("./test.txt","r");
  fread(buff,sizeof(buff),1,infile)
  printf("%s", buff);
  fclose(infile);
  return 0;
}
```

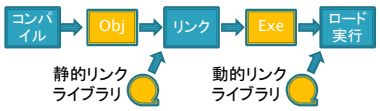
標準ライブラリ関数使用

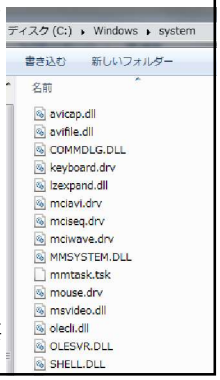
混在しないこと＝個別にバッファリング処理するため順序が不定となる
同理由で、stdioとiostreamの混在利用は不可（混在可の設定もあり）

12

ダイナミックリンクライブラリ

- Dynamic Link Library（ DLL、動的リンクライブラリ）：Windows等で採用
- Static Link Library(静的リンクライブラリ)：旧来の考え方





DLLの利点・難点
- 実行ファイルが小さくなる、メモリ領域が共用できる
- 互換性のあるDLLファイルが実行時に必要

13

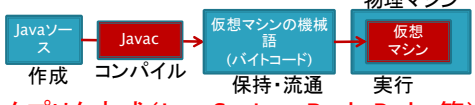
互換性(教科書3.4)

- 別システムへプログラムを移行するとき
- ソースプログラム互換性
  - 移行先の環境に合ったコンパイラで再コンパイルすれば動く
  - APIが同等、ANSI/ISO C規格
  - 整数ビット長・エンディングなどを考慮したコード
- オブジェクトプログラム互換性（バイナリ互換性）
  - そのままで動く、ハードウェアが同等
  - Windows用ソフト：別メーカのマシンでも動く

14

互換性

- 中間言語による互換性（Java言語）
  - 仮想マシンの機械語にコンパイル、仮想マシン上で実行
    ⇒一つのバイトコードがどこでも動く
    （仮想マシンのエミュレータがあれば）
    ＝ Write Once, run anywhere



- インタプリタ方式（JavaScript、Perl、Ruby等）
  - ソースプログラムのまま保持・流通、実行時に解釈・実行

15

移植性

- ソースプログラム互換性も実現困難
  ⇒できるだけ修正しやすく
- 移植性＝修正のしやすさ
  - 移植性が高いプログラムが望ましい
- 移植性を高めるには
  - 共通使用の範囲内の機能を使う（例：ANSI/ISO C規格）
  - マシン差を吸収する機能を使う（例ntoh,hton:バイト順の差吸収）
  - マシン分岐（コンパイル時、実行時）するコード（例#ifdef unix）
  - 標準機能、推奨機能を使う
  - 稼働実績の多い機能を使う
  - ・・・

ntoh (network to host)
hton (host to network)
ネットワーク標準のバイト順と当マシンでのバイト順を変換
（リトルエンディアンとビッグエンディアン）

16

本日の課題

1. APIとは何か。またそれが必要な理由を説明せよ。
2. 「互換性がある」と「移植性が高い」はそれぞれどういう意味か記せ。
3. （予習）OSの実行モードについて調べて記せ

- 今回のファイル名は“学籍番号-OS04.docx”
  (例：24238000-OS04.docx)としてください
- 締切：10月31日（金）18:00（遅れた場合は減点）

記載時の注意事項
- 参考資料(Webページ)がある場合は出典を書くこと
  - 出典を書かずに引用した場合は減点対象です

17

講義に関する注意事項

- 講義に関する連絡はLive Campusを用いて
  メールで行います
- 本講義に関する情報は次のWebpageに掲載するので、時々参照すること
  https://www.ai.is.saga-u.ac.jp/~hanada/OS/

注意：来週11月3日(月)は祝日です。
次回講義は6日(木)です。

18