

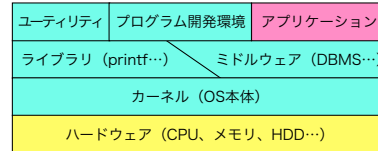
オペレーティングシステムの機能を使ってみよう 第1章 システムプログラミング

オペレーティングシステムの機能を使ってみよう

1 / 6

システムプログラムとは

- カーネル (OS の本体)
- ライブラリ (プログラムが使用するサブルーチン, DLL ...)
- ミドルウェア (DBMS, Web サーバ ...)
- ユーティリティ (ファイル操作, 時計, シェル, システム管理 ...)
- プログラム開発環境 (エディタ, コンパイラ, アセンブラ, リンカ, インタプリタ ...)



オペレーティングシステムの機能を使ってみよう

2 / 6

システムプログラミングとは

- システムプログラムを作成すること。
- 本講義ではユーティリティのプログラミングを行う。

なぜシステムプログラミング？

「システムプログラミングを通してオペレーティングシステムの体感的な理解」をする。

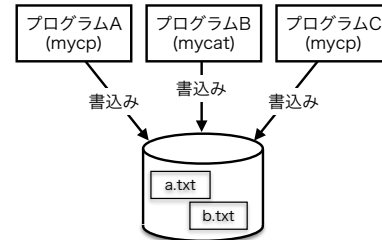
オペレーティングシステムを体感的に理解するために、オペレーティングシステムの機能を直接使用する簡単な CLI 版のユーティリティプログラムの作成 (プログラミング) を行う。

オペレーティングシステムの機能を使ってみよう

3 / 6

複数のプログラムが正しく実行されない

- コンピュータの中では複数のプログラムが同時に作動している。
- 各プログラムが勝手に資源にアクセスすると具合が悪い。
- 例えばディスクのどの領域をどのファイルが使用するか？
各プログラムが勝手に決めると不具合が起こる。

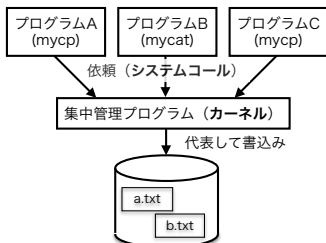


オペレーティングシステムの機能を使ってみよう

4 / 6

複数のプログラムが正しく実行される

- OS の本体 (カーネル) が代表して資源を管理する。
- 一般のプログラムはカーネルに処理を依頼し目的を達成する。
- 例えばファイルを作成してディスクのどの領域を使用するか？
カーネルが責任を持って一貫した管理を行う。(集中管理)
- 他のプログラムはシステムコールを行いカーネルに処理を依頼。



オペレーティングシステムの機能を使ってみよう

5 / 6

システムコールの使用

- C 言語からシステムコールを利用することができる。
- UNIX (macOS) の C 言語ではシステムコールと同じ名前の関数を呼び出すとシステムコールの発行になる。
- macOS 上で C 言語を用いてシステムコールを直接使用するユーティリティプログラムを作成する (システムプログラミングを行う)。
- OS の機能をシステムコールを通して実感する。
- OS が提供すべき機能を理解する。

```
// ディレクトリを作るユーティリティプログラム (mymkdir) の例
int main(int argc, char *argv[]) {
    if (argc!=2) {
        // エラー処理
    }
    mkdir(argv[1]); // ディレクトリを作るシステムコール
    return 0;
}
```

オペレーティングシステムの機能を使ってみよう

6 / 6