

オペレーティングシステムの機能を使ってみよう 第3章 高水準入出力と低水準入出力

オペレーティングシステムの機能を使ってみよう

1/9

高水準入出力と低水準入出力

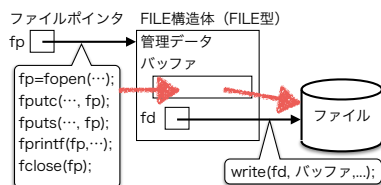
ファイルを読み書きするための機能
(API: Application Program Interface)

- 高水準入出力 (高水準 I/O)
豊富, かつ, 高機能な API
(`fprintf()`, `fscanf()`, `fputc()`, `fgetc()`, ...)
- 低水準入出力 (低水準 I/O)
システムコールのこと
少なく, かつ, シンプルな API
(`open()`, `read()`, `write()`, `lseek()`, `close()`)

オペレーティングシステムの機能を使ってみよう

2/9

高水準 I/O のデータ構造 (書き込み)

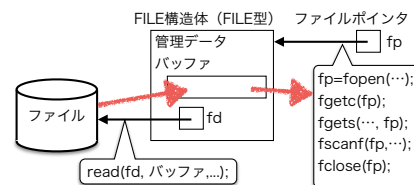


- ファイルポインタ (fp)
- FILE 構造体
- バッファリング
- write システムコール

オペレーティングシステムの機能を使ってみよう

3/9

高水準 I/O のデータ構造 (読み出し)



- ファイルポインタ (fp)
- FILE 構造体
- read システムコール
- バッファリング

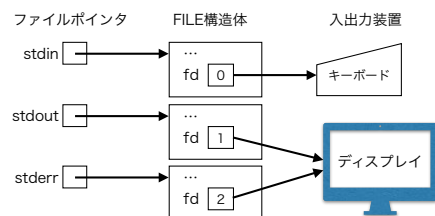
オペレーティングシステムの機能を使ってみよう

4/9

標準入出力 (標準入出力カストリーム)

名称	fd	fp	通常の接続先
標準入力カストリーム	0	stdin	キーボード
標準出力カストリーム	1	stdout	ディスプレイ
標準エラー出力カストリーム	2	stderr	ディスプレイ

fd: ファイルディスクリプタ
fp: ファイルポインタ



オペレーティングシステムの機能を使ってみよう

5/9

ユニファイド I/O

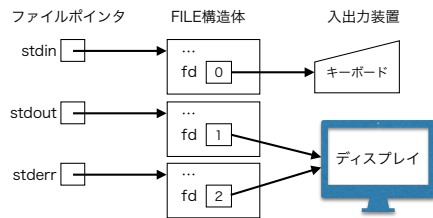
標準ストリーム	同じ意味の呼出し	役割
<code>scanf(...)</code>	<code>fscanf(stdin, ...)</code>	書式付きの入力
<code>getchar()</code>	<code>fgetc(stdin)</code>	1 文字入力
<code>-</code>	<code>fgets(stdin, ...)</code>	1 行入力
<code>printf(...)</code>	<code>fprintf(stdout, ...)</code>	書式付きの出力
<code>putchar(c)</code>	<code>fputc(c, stdout)</code>	1 文字出力
<code>puts(buf)</code>	<code>fputs(buf, stdout)</code>	1 行出力

- `printf(...)` と `fprintf(stdout, ...)` は同じ
- fp の代わりに `stdin`, `stdout` 等が使用できる.
- キーボードやディスプレイ (入出力装置) とファイルを同じ要領で操作できる.
- 入出力装置をファイルに統合 = (ユニファイド I/O)

オペレーティングシステムの機能を使ってみよう

6/9

標準入力ストリーム

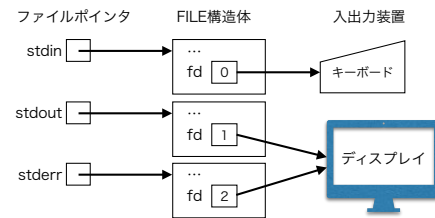


- ファイルポインタは `stdin`
- ファイルディスクリプタは 0 番
- ファイルディスクリプタ 0 は通常キーボードに接続
- ファイルポインタと FILE 構造体はプログラム起動時に初期化
- シェルはファイルディスクリプタ 0 をリダイレクト可能

オペレーティングシステムの機能を使ってみよう

7/9

標準出力ストリーム

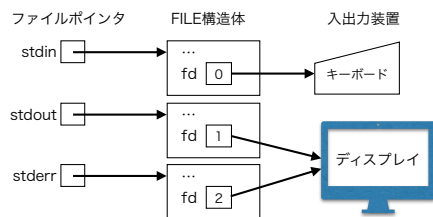


- ファイルポインタは `stdout`
- ファイルディスクリプタは 1 番
- ファイルディスクリプタ 1 は通常ディスプレイに接続
- ファイルポインタと FILE 構造体はプログラム起動時に初期化
- シェルはファイルディスクリプタ 1 をリダイレクト可能

オペレーティングシステムの機能を使ってみよう

8/9

標準エラー出力ストリーム



- エラーメッセージ出力用のストリーム
- ファイルポインタは `stderr`
- ファイルディスクリプタは 2 番
- ファイルディスクリプタ 2 は通常ディスプレイに接続
- ファイルポインタと FILE 構造体はプログラム起動時に初期化
- シェルはファイルディスクリプタ 2 をリダイレクト可能

オペレーティングシステムの機能を使ってみよう

9/9