ソフトウェア実習

3. ファイル入出力

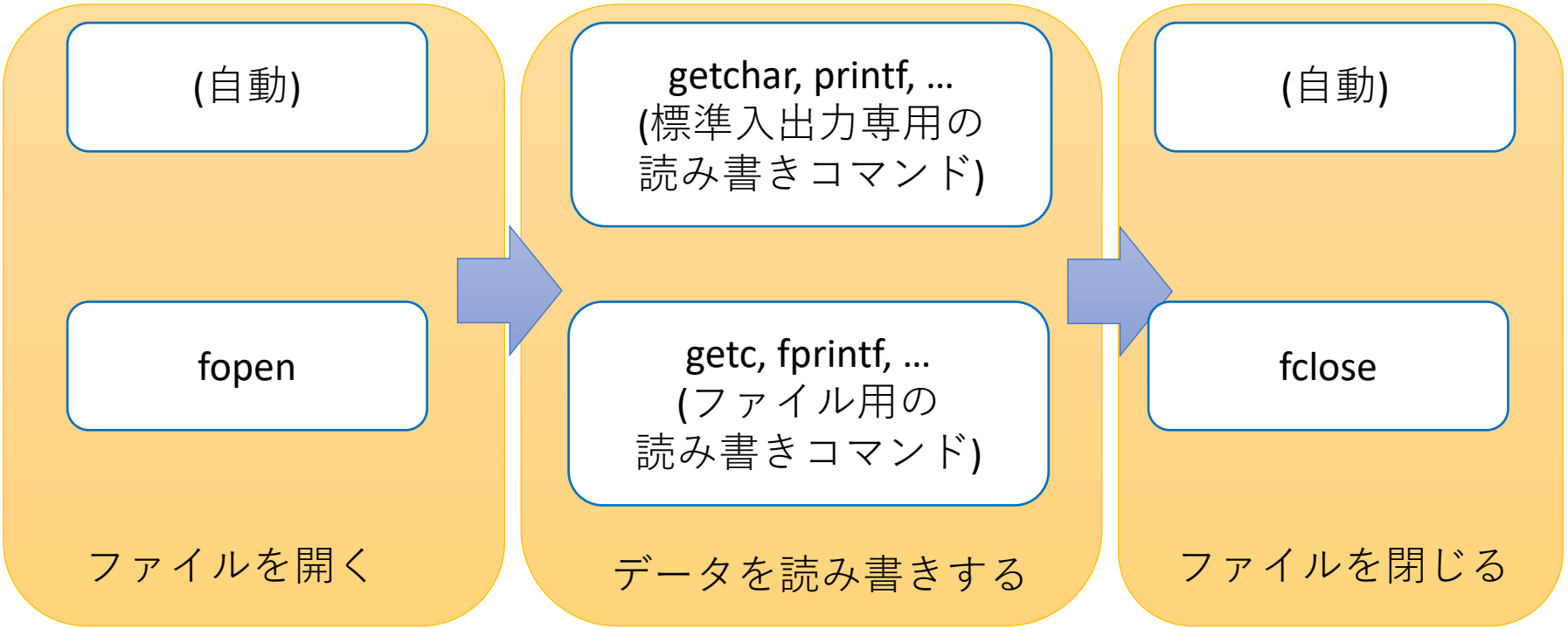
横山　大作

今回の狙い

- ファイル名を指定して読み書きする方法の復習

ファイル入出力の流れ

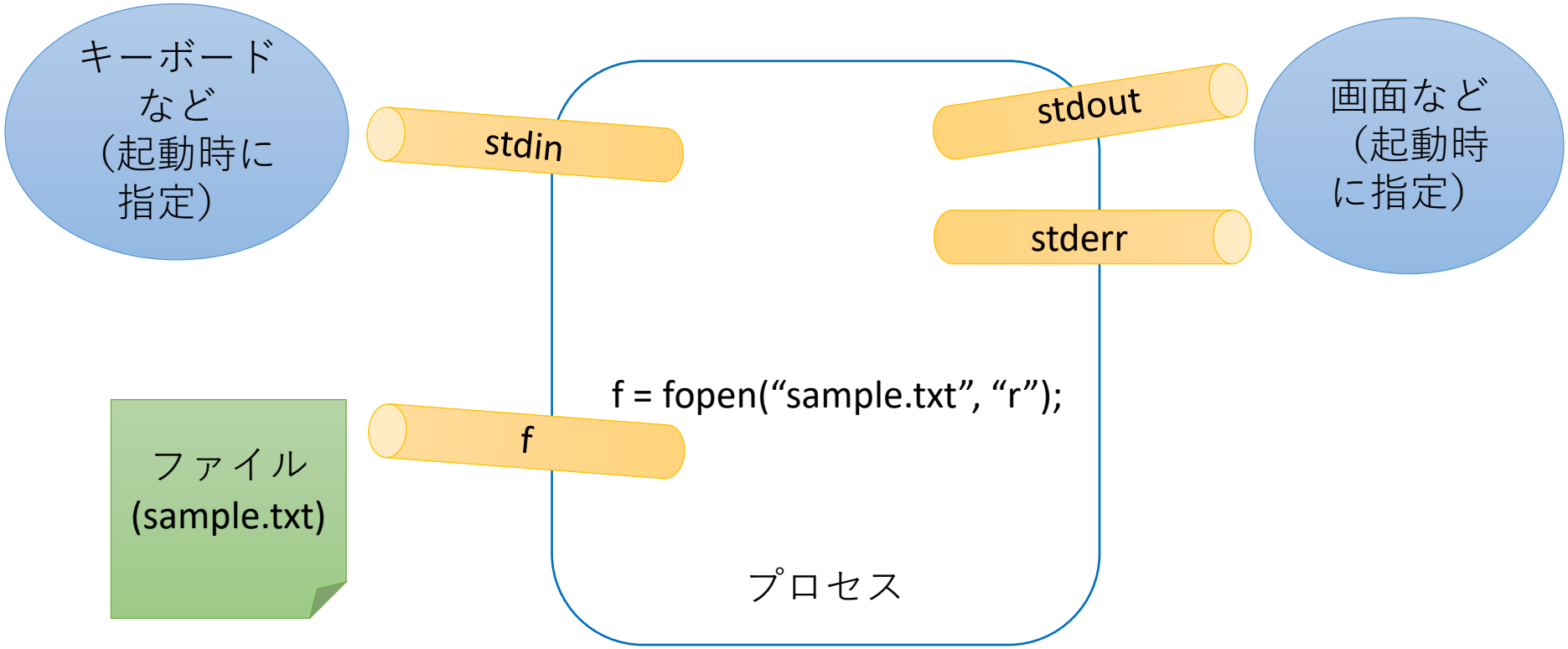
- 考え方はgetcharやprintfなどの、標準入出力を使う場合と同じ
  - ファイルを指定して読み書きするところだけが違う



プロセスと入出力

- プロセスは風船のようなもの
  - 外と隔てられた、独立した空間
- 入出力するときは出入り口を作る
  - ストローを刺すような感じ

プロセスとファイル


キーボード
など
（起動時に
指定）
stdin
stdout
stderr
画面など
（起動時
に指定）
f = fopen(“sample.txt”, “r”);
f
ファイル
(sample.txt)
プロセス

参考：mainの返り値

- プロセスから外の世界に情報を出す方法は（基本的に）2つ
  - ファイルで出す（標準出力も含めて）
    - これは今回の課題で練習している
  - 終了コードを出す
    - main()の返り値で、return 0; としているところ
    - この場合「終了コード0」で終了している
    - 実はプログラムを呼び出した側（普通はシェル）にこの値が伝わっている
      - bashの場合、$?という変数に入る
    - 終了させる関数、exit(0); の引数も同じ意味
  - UNIXのお約束として、返り値0は正常終了、!=0は異常終了

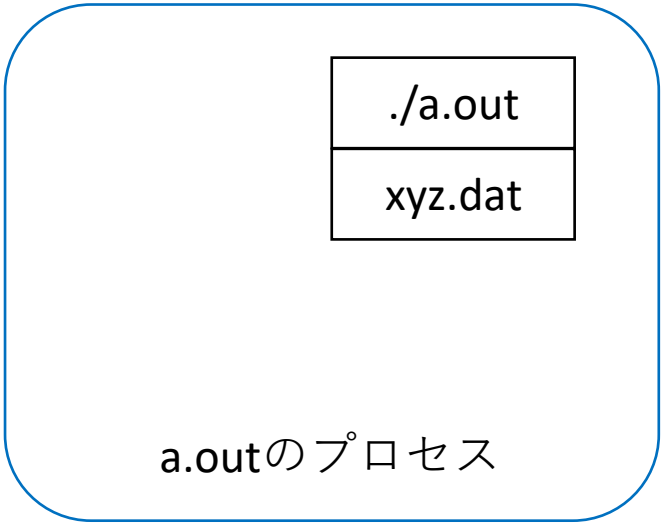
返り値の例

```
% ls
a.out  test.c  test.c~  xyz.dat
% echo $?
0
% ls hoge
ls: 'hoge' にアクセスできません: そのようなファイルやディレクトリはありません
% echo $?
2
```

- この返り値を使って、コマンドが失敗したら～する、みたいなことも書ける
  - 調べてみると良い

- 3.3 argc, argv
  - 使い方を理解しておきましょう
  - プログラムが始まるときに文字列を渡せるということです
  - 3.4の実例も見ましょう
  - おそらく、今後非常によく使います！
    - 実験データを処理するとき、データファイルを指定するとか

```
% ls
a.out  test.c  test.c~  xyz.dat
% ./a.out xyz.dat
```



参考: プロセスへの入力

- 外の世界からプロセス内への情報の入れ方は（基本的に）3通り

- ファイルで入れる（標準入力も含めて）
- 実行時の引数で渡す
  - main()のargc, argv
- 環境変数で渡す
  - (文字列,文字列)の辞書が使える
  - 興味がある人は調べてみましょう

- 3.4 ファイル入出力の実例
  - getc(f)がgetchar()相当
  - fopen, fcloseが追加されていますね
  - fprintf(stderr, …) はファイルへの出力で、出力先を標準エラー出力（自動的に開いてある）に指定したものです
- 3.5 – 3.8
  - それぞれの関数の説明
  - ファイルポインタがさっきの絵のパイプみたいなもの

入出力は副作用あり

- 入出力の関数が呼ばれるたびに「裏で何か起きてる」ことに注意
  - 入力: その文字はもう「読まれた」
  - ファイルポインタはただのポインタではなく、「どこまで読んだか」を覚えているようなデータ

```
while (getc(f) != EOF) {
    c = getc(f);


  ...
}
```

ここでのgetc()はもう違うデータを読もうとしている

この例に近い話が3.8 エラー誤りのチェックの説明で出てきてます。ぜひ考えてみてください。

対処の仕方

- 何らかの方法で返り値を覚えておく

```
int c;
c = getc(f);
while (c != EOF) {
  cを使った処理...


  ...
  c = getc(f);
}
```

- getcが2回出てきてかっこ悪い？見やすく書く方法もあります（が、少し理解を深めないと危険かも）
  - while ((c = getc(f)) != EOF) { … } よくわからないときは聞いて
- fscanfも同じような感じ（3.8章の問題）
  - fscanfの返り値が何なのかもよく理解しよう

- 3.9 改行文字
  - 画面では見えない文字が出力されている場合がある
    - 課題6とかで思った実行結果にならなくて困惑しがち
  - ファイルを「バイトとして」見られるようになっておこう
  - ついでに、2回目の講義の文字列の話も確認できますね
    - 文字コードは2回目にやった数値になっているかな？

課題

- 1〜6は1文字ずつ入力する練習
  - 1〜4はテキストのリストが参考になる
  - sample.txt, integers.txtなどのテスト用の入力データは自分で作ること。様々なケースがあるので、どういうファイルを入れたらテストになるかを考えて作ること。（レポートにどう工夫したか書こう）
  - 特に、課題6のテストは色々な場合が考えられるので良く確かめること
    - うっかり改行コードが違うファイルなどを作るとテスト結果がよくわからなくなる。確認するにはodコマンドを使うと良い
- 7,8は書式付きの入力の練習
  - fscanf
  - 入力ファイルをよく考えて作ってテストしよう
    - 「〜の可能性がある」をちゃんと読もう。課題8ではエラーとなる場合を「考える」ことが大きなテーマになっています。

注意

- 課題1,3,7の提出物は”sample.txt”のような教科書に指定されたファイル名でテストされます！
  - 間違えたファイル名を読み書きしてると実行テストが通らないので注意！
- それ以外の課題は、コマンドライン引数でファイル名を受け取ることが必要
  - 問題をよく確認して