プログラミング応用

Week6

本日の講義

- システムコールとは？
  - C言語のライブラリ関数とシステムコール
- システムコールの使い方
  - はじめてのシステムコール
  - システムコールによるファイル操作
  - コマンドライン引数

本日の講義

- システムコールとは？
  - C言語のライブラリ関数とシステムコール
- システムコールの使い方
  - はじめてのシステムコール
  - システムコールによるファイル操作
  - コマンドライン引数

復習：OSの内部構造

- OS　＝　カーネルの集合体
  - 各々のカーネルはOSの基本機能を提供
  - カーネルの提供する機能
    プロセス管理、空間管理、ファイル管理、
    割り込み制御、入出力制御、時間管理など
- アプリケーションソフトウェアは「システムコール」を使ってカーネルの機能を呼び出す
  - アプリケーションソフトウェアは基本的に
    ハードウェアを制御する命令を出すことは
    出来ない

システムコールとライブラリ関数

- システムコール
  - C言語からカーネルが提供する機能を使うときには必ずシステムコールを使うことになっている
  - open(), write(), close(), fork()などが代表的なシステムコール
  - C言語から関数として使うことが出来る
- ライブラリ関数
  - システムコールを組み合わせて何かしらのまとまった処理を実現する
  - printf(),scanf(),fopen(),fclose()などはライブラリ関数

子プロセスの生成などはライブラリ関数では行えない

straceコマンド(概要)

- straceコマンド
  - コマンド実行時にどのようなシステムコールが呼び出されているか確認するためのコマンド
    $ strace (実行ファイル名)

    例）strace ./a.out

straceコマンド(使い方)

- 以下のC言語プログラムをコンパイルし、straceコマンド付きで実行してみる
  $ strace ./a.out

```
1 #include <stdio.h>
2
3 int main(void) {
4    printf("This is a test.\n")
5 }
```

straceコマンド(実行例)

- $ strace ./a.outの出力例

```
mprotect(0x7f084ad54000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)      = 0
mprotect(0x7f084af81000, 4096, PROT_READ) = 0
munmap(0x7f084af6f000, 63466)            = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVA
, 0) = 0x7f084af7e000
write(1, "This is a test.\n", 16This is a test.
)        = 16
exit_group(0)                            = ?
+++ exited with 0 +++
```

mprotect()やfstat(), mmap(), write()などすべてシステムコール

本日の講義

- システムコールとは？
  - C言語のライブラリ関数とシステムコール
- システムコールの使い方
  - はじめてのシステムコール
  - システムコールによるファイル操作
  - コマンドライン引数

UNIXのシステムコール

- UNIXの用意しているシステムコール
  - 約300種類
  - 本講義では代表的な以下の6つについて使い方を学ぶ
    - ファイルシステムを扱う
      - open()
      - read()
      - unlink()
      - close()
    - 出力を行う
      - write()
    - 子プロセスを生成
      - fork()

write()関数の使い方

write(ファイル記述子, バッファ, バッファ長);

ファイル記述子
　整数1、2，もしくはファイルディスクリプタ数を指定
　1：標準出力
　2：　標準エラー出力
＊ファイルディスクリプタ数については後ほど説明

バッファ
　文字列を格納したポインタ変数
バッファ長
　文字列の長さ（sizeof()関数で取得できる）

write()関数のサンプルプログラム

- システムコールwrite()関数を使って文字列を表示

```
1 #include <stdio.h>
2
3 int main(void) {
4     char* s = "Sample Sentence";
5     write(0, s, sizeof(s));
6     return 0;
7 }
```

本日の講義

- システムコールとは？
  - C言語のライブラリ関数とシステムコール
- システムコールの使い方
  - はじめてのシステムコール
  - システムコールによるファイル操作
  - コマンドライン引数

open()関数の使い方

- ライブラリ関数のfopen()とほぼ同等の動作

open(“ファイルへのパス”, モード);

モードの指定方法:

O_RDONLY	**読み込み専用(“r”)**
O_WRONLY	書き込み専用(“w”)
O_RDWR	読み書き可能(“w+”)

close()関数の使い方

- ライブラリ関数のfclose()とほぼ同等の動作

close(ファイルディスクリプタ);

ファイルディスクリプタ

- open()関数の戻り値はファイルディスクリプタという整数値
- ファイルディスクリプタ
  - openしたファイルを一意に特定するための番号
  - 整数3以降が返る
  - 0, 1, 2はすでに予約済み
    0: 標準入力
    1: 標準出力
    2: 標準エラー出力

サンプルプログラム(open, close)

```
 1 #include <stdio.h>
 2 #include <fcntl.h>
 3
 4 int main(void) {
 5    int file1 = open("sample1.txt", O_RDONLY);
 6    int file2 = open("sample2.txt", O_RDONLY);
 7    printf("ディスクリプタ：%d\n", file1);
 8    printf("ディスクリプタ：%d\n", file2);
 9    close(file1);
10    close(file2);
11    return 0;
12 }
```

```
$ ./a.out
ファイルディスクリプタ:3
ファイルディスクリプタ:4
```

read()関数の使い方

- openしたファイルの中身を読み込み
  バッファに格納

read(ファイルディスクリプタ, バッファ, バッファ長);

  - バッファ：文字列を格納する配列
    (例えばchar buffer[256];などと定義)
  - バッファ長：バッファの長さ
    （sizeof(buffer);等で取得）

サンプルプログラム(read)

- sample1.txtの中身が表示される

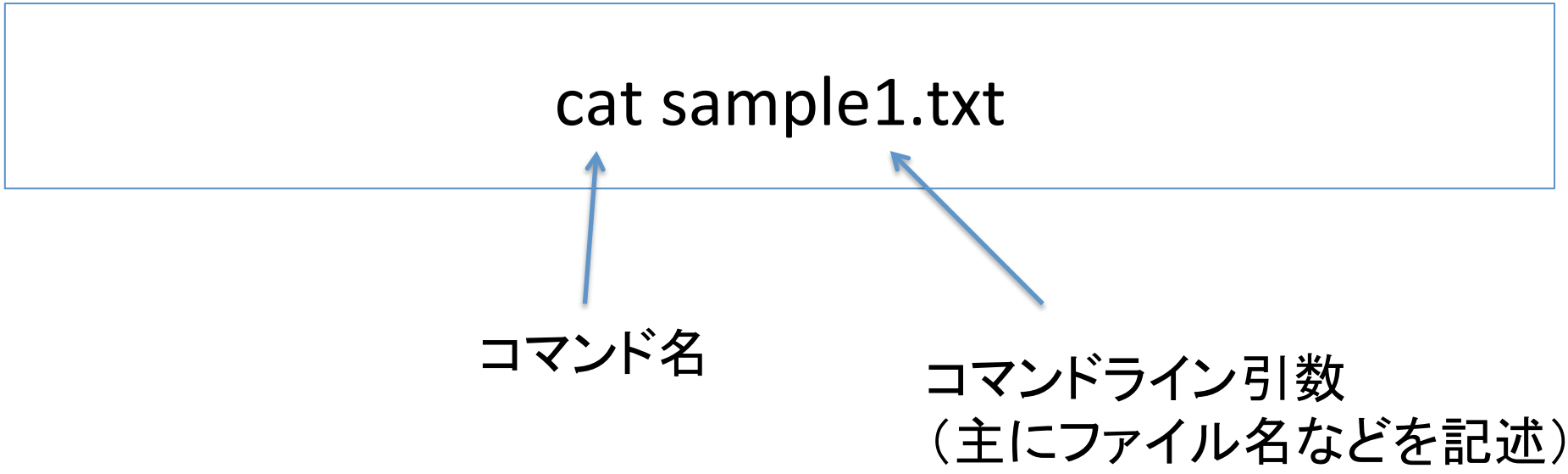
```
 1 #include <stdio.h>
 2 #include <fcntl.h>
 3 #include <unistd.h>
 4
 5 int main(void) {
 6   char buffer[256];
 7   int file = open("sample1.txt", O_RDONLY);
 8   read(file, buffer, sizeof(buffer));
 9   printf("%s", buffer);
10   close(file);
11   return 0;
12 }
```

本日の講義

- システムコールとは？
  - C言語のライブラリ関数とシステムコール
- システムコールの使い方
  - はじめてのシステムコール
  - システムコールによるファイル操作
  - コマンドライン引数

コマンドライン引数

- catやgrepなどのUNIXコマンドもC言語で記述されている



C言語プログラムとコマンドライン引数

- C言語からコマンドライン引数を取得する方法
  - int main(void)を以下のように変更
    int main(int argc, char *argv[])
  - argc: 引数の数
  - argv[0]: コマンド名
    argv[x]: x番目の引数を文字列として取得可能

サンプルプログラム(コマンドライン引数)

```c
#include <stdio.h>

int main( int argc, char *argv[] ) {
  printf("引数の数:%d", argc);
  printf("コマンド名：%s", argv[0]);
  printf("ひとつ目の引数：%s", argv[1]);
}
```

```
$ ./a.out sample.txt
引数の数:2
コマンド名 : ./a.out
ひとつ目の引数 : sample.txt
```

本日の演習

- システムコールを用い、ファイルを移動するプログラムを作成
- すべて終わるとUNIXのmvコマンドに近い動作をするmymvというコマンドが出来上がる

演習0(準備)

- 以下のコマンドでローカルのGitリポジトリに移動して作業を開始
  $ cd c-programming-practice-(学籍番号)

- ブランチを切って移動
  $ git branch system_call
  $ git checkout system_call

- 作業用ディレクトリを作成し移動。sample.txtというファイルを作成し、ファイルにtestと書き込む
  $ mkdir week6
  $ cd week6
  $ echo “test” > sample.txt

演習1(コマンドライン引数)

1. week6ディレクトリ以下にmv_command_library.cという空のファイルを作成
2. sample.cというファイルをコマンドライン引数として受け取り、printf()関数でファイル名を出力しなさい

演習2-1（標準ライブラリによるmv）

- システムコールではなく、標準ライブラリを用いてsample1.txtの内容をsample2.txtにコピーするプログラムを記述しなさい
（2年生のプログラミング基礎の資料を参照）

演習2-2（straceコマンド）

- 演習2-2で作ったプログラムをstraceコマンド付きで実行しなさい
  - $ strace ./a.out
- どのようなシステムコールが呼び出されているか考察しなさい

演習3-1（システムコールによるmv）

- mv_command_systemcall.cというファイルを作成せよ
- コマンドライン引数として2つのファイル名を受け取りprintf()で表示するプログラムを作成せよ

演習3-2（システムコールによるmv）

- 第1引数として受け取ったファイルを読み込み専用でopenするシステムコールを追加せよ

- 第2引数として受け取ったファイルを書き込み可能で、かつ、ファイルがすでに存在する場合には空にするシステムコールを追加せよ
  - ヒント:
    読み書き可能でかつ、ファイルが存在するならば空にするには以下の記述を用いる
    open(ファイル名,O_WRONLY|O_CREAT|O_TRUNC,0666);

演習3-3（システムコールによるmv）

- 第1引数で受け取ったファイルをオープンし、第2引数で受け取ったファイルに書き込みなさい
  - ヒント：
    以下の関数を使うと良い
    open()関数
    write()関数

演習3-4（システムコールによるmv）

- 第1引数で受け取ったファイルを削除せよ
  - remove()関数もしくはunlink()関数の使い方を各自調べて、記述せよ。

演習3-5（システムコールによるmv）

- 以下の2つのファイルをクローズし、プログラムを終了せよ
  - 第1引数で受け取ったファイル
  - 第2引数で受け取ったファイル

演習4-1（早く終わった人向け）

- 演習で作成したmvコマンドのような動作をするプログラムを、以下の手順で実際にコマンドとして呼び出せるようにしてみましょう
  1. 以下のコマンドでbashの設定ファイルを開く
     $ gedit ~/.bashrc
  2. 以下を記述
     PATH=$PATH:~/c-programming-practice-(学籍番号)/week6
  3. 端末で以下のコマンを入力し、設定ファイルを再度読み込む
     $ source ~/.bashrc

演習4-2（早く終わった人向け）

- mv_command_systemcall.cを以下のコマンドで”名前付きコンパイル”

  $ cc -o mymv mv_command_systemcall.c

  ※-oオプションを付けるとコンパイル後に出来る実行ファイルに任意の名前を付けられる。
  この場合mymvという実行ファイルが出来る。

演習4-3(早く終わった人向け)

- mymvコマンドが正しく動くことを確認
  - $ mymv sample1.txt sample2.txt
- sample1.txtがsample2.txtという名前に変わっていれば成功
- 早く終わった場合には、catコマンド、echoコマンド、cpコマンド、grepコマンドなどの実装にも挑戦してみましょう。

次回

- 試験前の演習回になります
- 試験についての説明をするので必ず出席してください