Cプログラミング1(再)
第5回

講義では、Cプログラミングの基本を学び
演習では、やや実践的なプログラミングを通して学ぶ

Cに必要なコンピュータ知識

Cはコンピュータの力を引き出せるように設計

コンピュータの知識が必要

コンピュータの構造

1. パーソナルコンピュータの構造

自分の(目の前にある)コンピュータの仕様を調べてみよう

パソコン本体= CPU(中央処理装置):

制御装置、演算装置、レジスタ、キャッシュメモリ

補助記憶装置(ハードディスク)

メモリ(主記憶装置)

入力装置:キーボード、マウス、スキャナ等

出力装置: モニタ(ディスプレイ)、プリンタ等

2．メモリにはアドレスが付いている

数の接頭語:　k　M　　G　　T

アドレス(address, 番地) : メモリに付けられた番号
　16進数で表現される : 0x000000 ～ 0xFFFFFF

リソース(resource, 資源): メモリ、ハードディスクの記憶容量など

3．メモリへの読み書き

CPUでの計算

レジスタ：電卓の「メモリ」(記憶場所)に相当

主記憶からデータを「アドレス」によって取り出す

→ レジスタにデータを取り込む（読み込み）

→計算する（計算結果はレジスタに残る）

→レジスタの値をメモリに書き込む（書き込むにもアドレスが必要）

プログラムは「メモリの特定のアドレスに格納されているデータに対して処理を行う」ものである(p.152)

4．CPUとメモリはバスでつながっている

バス(bus) : いろいろなデータが行き来する道

CPUとメモリをつなぐバス

- データバス　　読み書きするデータのやりとり
- アドレスバス　読み書きするアドレスの指定
- コントロールバス　データを書き込むのか読みこむのかを指定

１つの信号線は１ビットのデータを送る

アドレス空間：アドレスバスの本数＝指定できるアドレスの大きさ

5．ROMとRAM

ROM: Read Only Memory

読み込み専用のメモリ

RAM: Random Access Memory

読み書き両用のメモリ

6．CPUと入出力装置もバスでつながる

基本的に、入出力装置もメモリ同様バスで接続され、メモリと同様にアドレスが付けられている＝メモリマップドI/O

メモリとは異なる入出力専用のバスが用意されていることもある

プログラムが実行されるまで

1. CPUを構成する３つの装置

制御装置　　CPUの処理内容を決める
　　制御信号を送る、セレクタへ信号を送る

演算装置(ALU)
　　足し算や引き算を行う
　　演算はレジスタのデータだけが対象

レジスタ群
　　命令やデータの一時記憶
　　アキュームレータ、汎用レジスタ、インデックスレジスタ、プログラムカウンタ、命令レジスタ、フラグレジスタ、スタックポインタ、浮動小数点数演算レジスタ

2. CPUの基本処理

- 読み込み

  メモリの特定の番地の値を特定のレジスタに記憶

- 書き込み

  特定のレジスタの値をメモリの特定の番地に記憶

- 演算処理

  特定のレジスタと特定のレジスタの間で演算し、結果を特定のレジスタに記憶

3. プログラムはメモリ上に置かれる

インストール(install、「実装」)

プログラムを実行できるように準備すること

注: 似た言葉で「ダウンロード(download)」があるが、プログラムをダウンロードしただけでは使えない

プログラムは通常ハードディスクに保存されている

それを「実行」するにはメモリ上に配置する＝ロード(load)

プログラムの実行=run, execute

4. マシン語命令の実行

CPUがプログラムを実行する時のサイクル

1) fetch (フェッチ、命令の取り出し）
2) decode(デコード、解読）
3) operand fetch(オペランド・フェッチ、処理対象の取り出し)
4) execute(エグゼキュート、命令の実行)
5) write back(ライトバック、結果の保存)

プログラムの進行

プログラムもメモリに置かれている

プログラムカウンタによって「どのプログラムが実行されているか」を管理

基本的には、プログラムがフェッチされたら、カウンタの数が１命令分だけ増える＝順次処理

反復処理：プログラムカウンタの値を小さな値に戻す

選択処理：プログラムカウンタの値を先に進める（途中の処理を飛ばす）

ライブラリの実行

プログラムにおいて、別個に用意されているプログラム（ライブラリ）を実行する場合

1)今のプログラム・カウンタの値をスタックに保存

2)プログラムカウンタの値をライブラリの先頭のアドレスにセット

⇒ そのライブラリが実行される

3) ライブラリの実行が終わったら、スタックからプログラムのアドレスを取り出す

4) その次のプログラムから実行を行う

5．演算装置(ALU)による演算処理

ALU : Arithmetic Logic Unit

2つの数値の入力からひとつの演算結果を得る

入力はレジスタに入っている

出力もレジスタに入っている

「レジスタAの値を 3000番地に保存する」には、

3000を別なレジスタBに保存して、レジスタAの値をBの値となっている番地のメモリに書く

事が必要

ポインタ

ALUがもたらす情報

演算結果の出力だけではなく、
フラグ情報も出力　--- 演算結果が0か、マイナスか、
桁あふれが生じたか、偶数か奇数か

フラグレジスタにフラグ情報が格納され得る

6．メモリバスのビット数

アドレスバスの大きさ

メモリの容量が決まる---アドレスでメモリの番地を指定

8本 なら 2^8 通りの番地⇒256

16本 なら 2^{16} 通りの番地⇒65536

…

汎用レジスタの大きさ＝データバスの大きさ

CPU性能：32ビットCPUなら　32ビット(4バイト)

64ビットCPUなら　64ビット(8バイト)

一度にやりとりできるデータの大きさ⇒int型のビット長

プログラムの構造

1.　プログラムの種類

1) デバイスドライバ(device driver)

コンピュータを構成するハードウェアを動かす

入力デバイス: キーボード、マウスなど

出力デバイス: プリンタ、サウンド、モニタなど

新しい機器を接続するにはそのデバイスドライバを組み込む必要がある=インストール(install)

プログラムの構造

2) アプリケーション(application)

「応用ソフト」　　ワープロ、ゲームなど

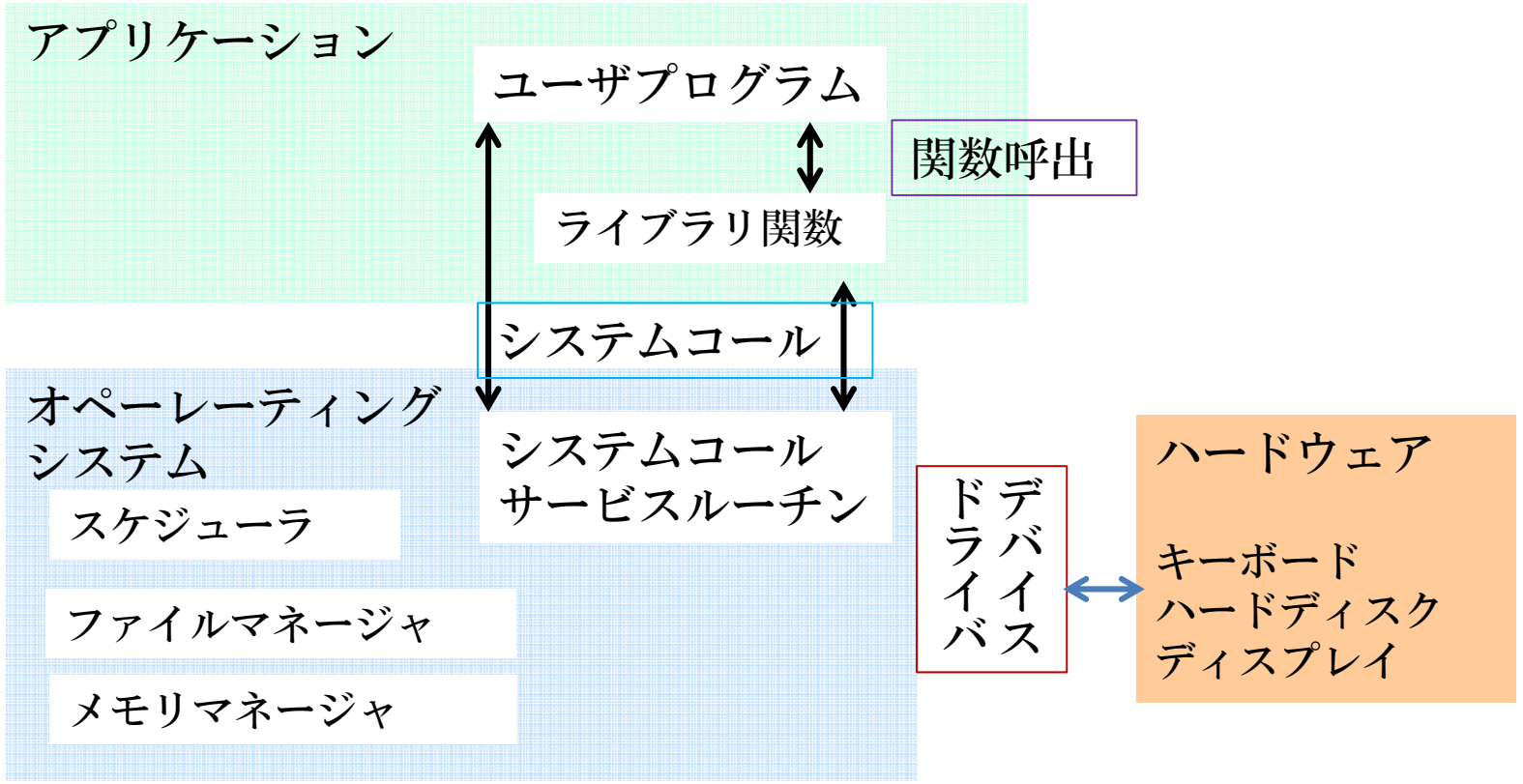
3) オペレーティングシステム(OS)　「基本ソフト」

コンピュータシステム全体の管理システム

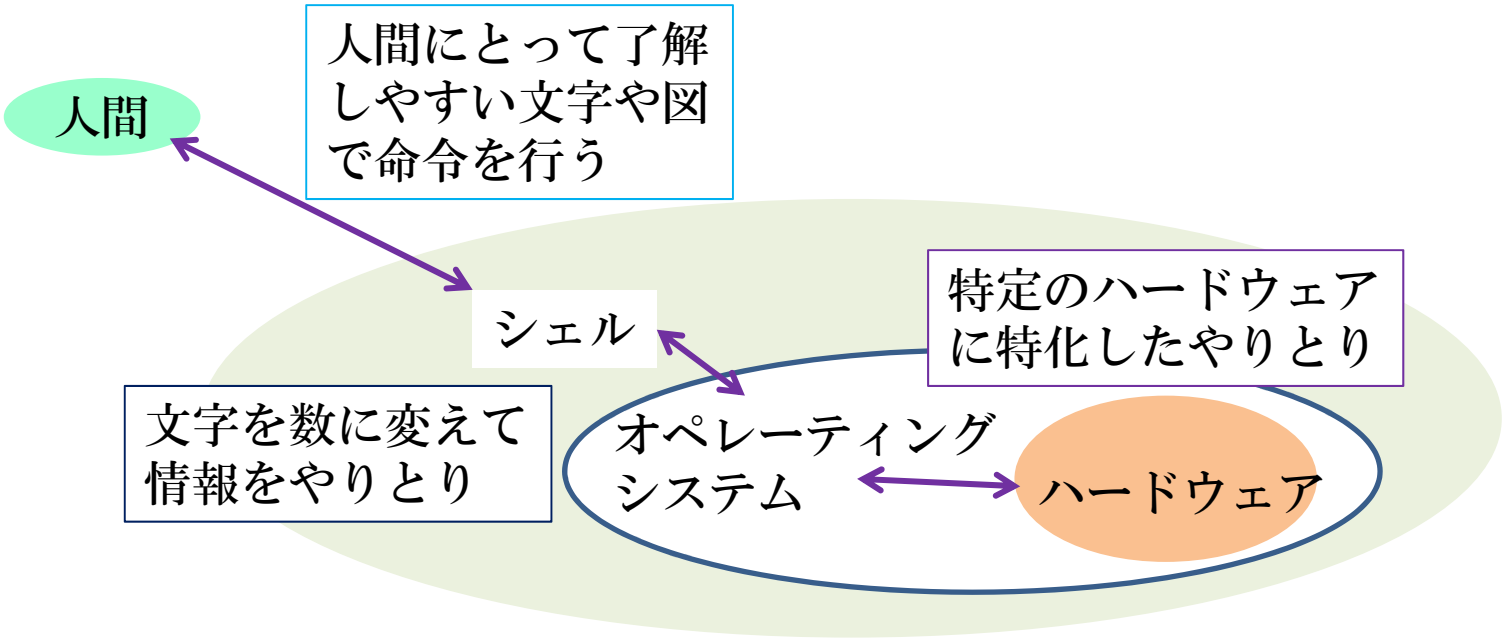
Windows, MacOS, Linux, Unix, Android, iOS, ...

CはUnix開発のために作られたプログラミング言語

2．ソフトウェアの階層構造



3. プログラムの実行とシェル



4. プログラムの構造

実行可能プログラム(binary)は、セグメントに分けられてメモリ上にロードされ、実行される

テキストセグメント: マシン語命令の格納領域
ユーザプログラムやライブラリなど

データセグメント: 作業対象情報の格納
読み込み専用、静的領域、動的領域

スタックセグメント: 情報の一時的記憶用
自動変数、プログラムカウンタ

5. プロセスとスレッド

5.2.4節から: CPUによるプログラムの実行

(1)命令の取り出し(フェッチ), (2)デコード(解読),

(3) オペランドフェッチ、(4)エグゼキュート(命令実行)

(5) ライトバック(結果の保存)

プログラムを実行させるのはオペレーティングシステム

プロセス、スレッド: CPUがプログラムを実行する単位

高機能なシステムなら マルチプロセス、マルチスレッド

低級なシステムの例: Arduino --- シングルプロセス

プロセスとスレッド(続)

一つのCPUしかないコンピュータでも「見かけ上マルチプロセスで走る」

タイムシェアリング(TSS): 短い時間間隔でプロセスを切り替えながら実行する

プロセスとスレッドの違い: メモリ空間が同じかどうか

異なる	共有

入出力

1. 入出力とバッファ
   バッファとは

2.標準入出力
   標 準入出力とは　　自分で説明を試みよ
   Cの標準ライブラリ

3. リダイレクト

リダイレクト(redirect) = re + direct

UnixやWindowsではリダイレクトの記号がある
実際に試してみよう:
実行プログラム名 < 入力ファイル名
実行プログラム名 > 出力ファイル名
実行プログラム名 >> 出力ファイル名
実行プログラム名 < 入力ファイル名 > 出力ファイル名

4. ファイル入出力

ファイルに保存されたデータの読み込みや、ファイルにデータを書き込む処理の手順

1) ファイル名を指定して、ファイルを開く
2) ファイルを読み書きする
3) ファイルを閉じる

Cプログラミング

0) ファイル操作の変数を用意
例:
```
FILE   *fp;
```

1) fp = fopen(“ファイル名”,”r”)
```
fp = fopen((“ファイル名”,”w”)
```

2) fscanf(fp, “%d”, &x);
```
fprintf(fp, “%d = %d”, a,b);
```

3) fclose(fp)

5. ストリーム

ストリーム(stream)　　入出力を表す

　　元の意味は「流れ」---データの流れ

バイトストリーム

　　テキストデータ vs バイナリデータ

6. 入力の終わりはEOF

EOF = End Of File　ファイルの終わり

これは制御コード

入力するには Ctrl D (Unix)

Ctrl Z (Windows)

ただしこれは「シェル」によって伝わる

宿題

(1) リダイレクトを使って、プログラムの出力結果をファイル result.txt に書き出してみよう

(2)リダイレクトを使って、本来はキーボードからの入力をファイルから入力させてみよう

(3)ファイルを読み込み、その結果を逆順にして別なファイルreverse.txtに書き出すプログラムを書こう