# 物理学情報処理演習

2015年5月1日

ver20140521(コマンド集update)

## 4. 計算機、UNIX、C言語

身内賢太朗

レポート提出：fsci-phys-jouhou@edu.kobe-u.ac.jp

# 4.1　計算機

# 4.1.1　計算機の基本構成

■処理装置：processor

情報を***演算・制御***

“命令”に従って、”データ”の処理を行う

■(主)記憶装置：memory

情報を***記憶***

命令、データとも同一の扱い

■入出力装置：I/O device

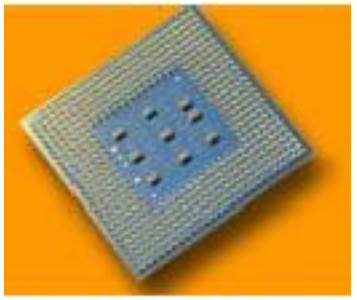
情報を外部から***入力***、外部へ***出力***

- ♦キーボード
- ♦ディスプレイ
- ♦ファイルシステム
- ♦ネットワーク

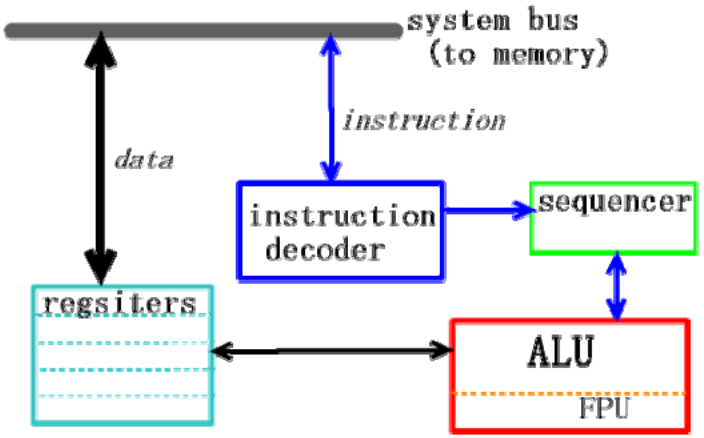
**→これらの装置によりコンピュータが動作する**

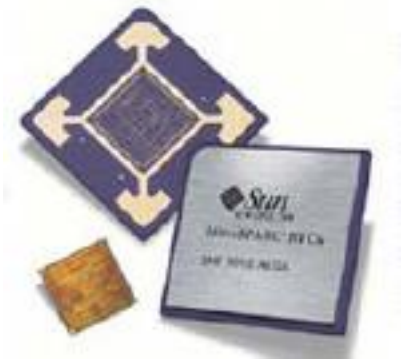
# Processor

- CPU Central Processing Unit（中央処理装置）
  記憶装置から命令を取り込み、それに応じて記憶装置上の情報を処理
  - 論理演算装置：ALU (Arithmetic Logic Unit)
  - 汎用レジスタ：register（プロセッサ内部にある、演算や実行状態の保持に用いる記憶素子）
  - シーケンサー：sequencer（順番を制御するコントローラー）
  - 命令解読器：instruction decoder



Intel Pentium4

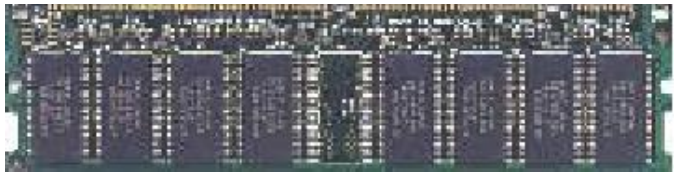


SUN Ultra SPARC III

# Memory：主記憶装置

- メモリー：計算機上で情報を保存する主な領域
  - 現在、主記憶は半導体素子による記憶媒体で実現されている



- 計算機上では、情報は2進数(ON/OFF)で扱われる
  - Bit: 0 or 1
  - Byte: 8 bit 0 ~ 255
    - Ex) 58(0x3A) = 00111010
    - 通常、計算機で扱える最小単位
  - Word: 16 bit = 2 byte 0 ~ 65535
  - Long Word: 32 bit = 4 byte 0 ~ 42949672965

# Memory: 2進数, 10進数, 16進数

■2進数、10進数、16進数の関係

2進数4桁を考える

0000 = 0 (10進数) = 0 (HEX)
0001 = 1 (10進数) = 1 (HEX)
0010 = 2 (10進数) = 2 (HEX)
・・・・ ・・・・
1001 = 9 (10進数) = 9 (HEX)
1010 = 10 (10進数) = A (HEX)
1011 = 11 (10進数) = B (HEX)
・・・・ ・・・・
1110 = 14 (10進数) = E (HEX)
1111 = 15 (10進数) = F (HEX)

2進数8桁を考える(1 byte)

11111111 = 255 (10進数) = FF (HEX)
1 byte は 16進数 2桁で表現可能

# Memory

■Address:　アドレス

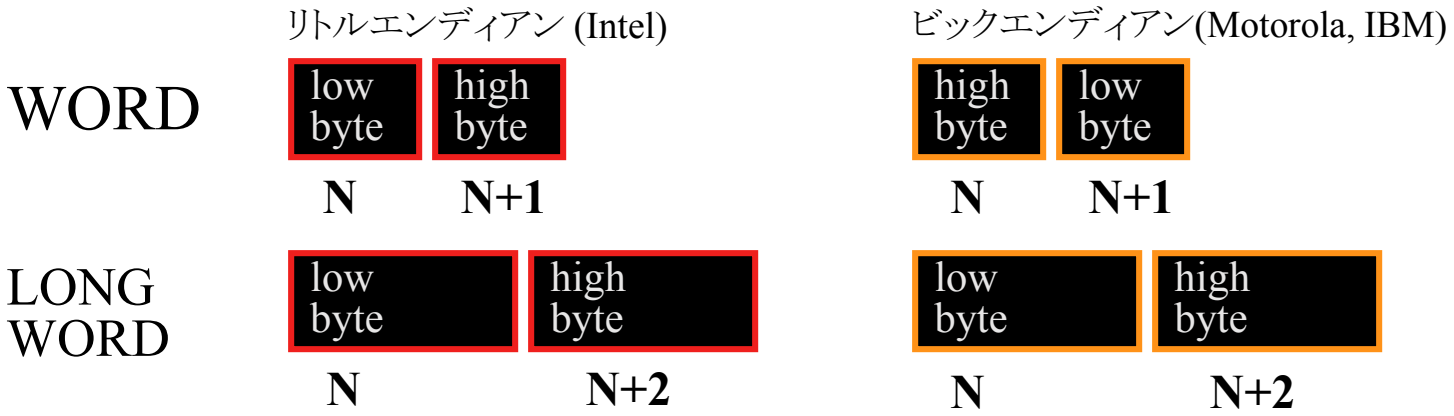
◆メモリーの特定の領域を指定するものをアドレスと呼ぶ

♦64K bytes (64 x 1024 byte)　= 16 bit address

♦1M bytes (1024 K bytes)　= 20 bit address

♦4G bytes (4 x 1024 M bytes) = 32 bit address

◆通常、あるアドレスはByte(もしくはそのByteから始まる連続領域)を指す



# Memory model

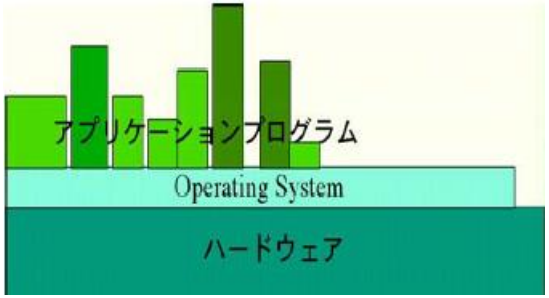
- メモリーは計算機のプログラムとデータを保存する領域であり、それらをプログラムの上でどのように見るかは重要
- データ保存階層
  - 汎用レジスタ
  - キャッシュメモリー
  - 主メモリー
  - 二次記憶
- 仮想メモリーシステム
  - プログラム上のアドレス（論理アドレス）をハードウエア上のメモリーアドレス（物理アドレス）に投影する機構
  - これを実装するにはハードディスクやネットワーク（でマウントされた他の計算機のディスク）など二次記憶装置が必要

# I/O device: 入出力

- ■外部との情報のやりとりを行う機器
  - ◆外部
    - ♦ユーザー
    - ♦他の計算機
    - ♦周辺機器
  - ◆入出力機器
    - ♦ユーザーインターフェース
      - キーボード
      - マウス
      - ディスプレイ
    - ♦補助記憶（二次記憶）
      - ハードディスク
      - CD-ROMドライブ
    - ♦周辺機器
      - ネットワーク
      - プリンタ
      - 計測機器

# 4.1.2 Operating System (OS)

- OSとはコンピュータを動作させるための基本ソフトウエア
- OSの役割
  - 計算機資源を有効に運用するためのソフトエアの集合
  - アプリケーションから計算機資源を使用する際のインターフェースを提供
    - 計算機資源とは
      - CPU
      - 主記憶
      - 入出力装置
      - タイマー　　等々




- UNIXは、ワークステーションの汎用OS
  - パーソナルコンピューター用のOSにはMS-DOS Windows, MacOS X などがある。
  - (広義の)UNIXには、色々な種類がある
    - System-V系　HP-UX, IRIX (Silicon Graphix)
    - BSD系　　SUN OS, Solaris(SUN), FreeBSD, NetBSD
    - 独立系　　Linux, AIX(IBM)

# Operating System (OS)

## ■UNIX

### ◆ユーザーの管理

- ♦UNIXは複数ユーザー使用を前提としているためlogin管理などを行う

### ◆アプリケーションプログラムなどのジョブ管理

- ♦UNIXは複数のプログラムが走っている。それらのプログラムをスムーズに走らせるためのメモリーやCPU管理を行う。

### ◆アプリケーションプログラムからの入出力処理の要求の処理

- ♦アプリケーションプログラムがハードウエアに対して要求する入出力に応じてハードウエアを制御する。

### ◆ネットワーク管理

- ♦UNIXのネットワークはEthernet規格やISDNなどの回線による方法でネットワークを構築できる。これらやメール送受信などを管理する。

### ◆その他ハードウエア、ソフトウエアに関する管理

# 4.1.3　プログラミング言語

■プログラミング言語とは

- ♦ 計算機の処理の手順を規定するもの

■機械語

- ♦ CPUが直接理解できる二進法で表された形式

■アセンブリー言語

- ♦ 機械語と一対一対応づけられた、人間が理解できる形式

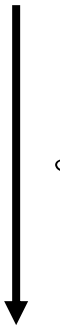
■高級言語

- ♦ 処理手順を論理的に示した、ハードウエアに直接は依存しない形式の言語

# Programming Language

■用途による分類
- ◆制御　　Assembler, C, C++
- ◆数値計算　FORTRAN, C
- ◆事務処理　COBOL, Basic, C++
- ◆データーベース SQL
- ◆Web　　HTML, JAVA
- ◆教育　　Basic, Pascal

■機能分類
- ◆Assembler
- ◆C
- ◆FORTRAN, Basic
- ◆JAVA, C++



より抽象化

# Compiler and Interpreter

- 高級言語で書かれたプログラムを機械語に翻訳する必要がある
  - インタプリタ型言語
    - プログラムを一行ずつ解釈し、CPUに命令して実行
    - Basic, UNIXシェル, AWK, perl
      - 長所：プログラムが完全なものでなくても実行可能。逐次動作なのでエラー処理に優れる
      - 短所：処理速度は遅い
    - 小規模プログラムに適する
  - コンパイラ型言語
    - プログラムをコンパイラによって機械語に翻訳し、リンカーで処理に必要な機械語を実行ファイルにまとめてから実行
    - FORTRAN, C, C++
      - 長所：プログラムの実行速度が速い。部分翻訳も可能。
      - 短所：文法、論理的にも完全なプログラムが必要

# Compiler

- ■プリプロセッサ：pre-processor
  - ◆前処理として、ソース・コード(source code)を整形
- ■コンパイラー：compiler
  - ◆プログラミング言語を実行単位毎に（例：関数）解読して、機械語に翻訳し、オブジェクト・ファイル（object file）を作る
- ■リンカー：linker
  - ◆プログラムに関するオブジェクト・ファイルやライブラリーをまとめて、実行可能イメージにする
    - ♦ライブラリー：library
      - Object fileを機能単位でまとめたもの
    - ♦実行可能イメージ：load module, executable file
      - いわゆるプログラムのこと。主記憶に導入され、実行点から逐次実行される。

# プログラム作成から実行まで

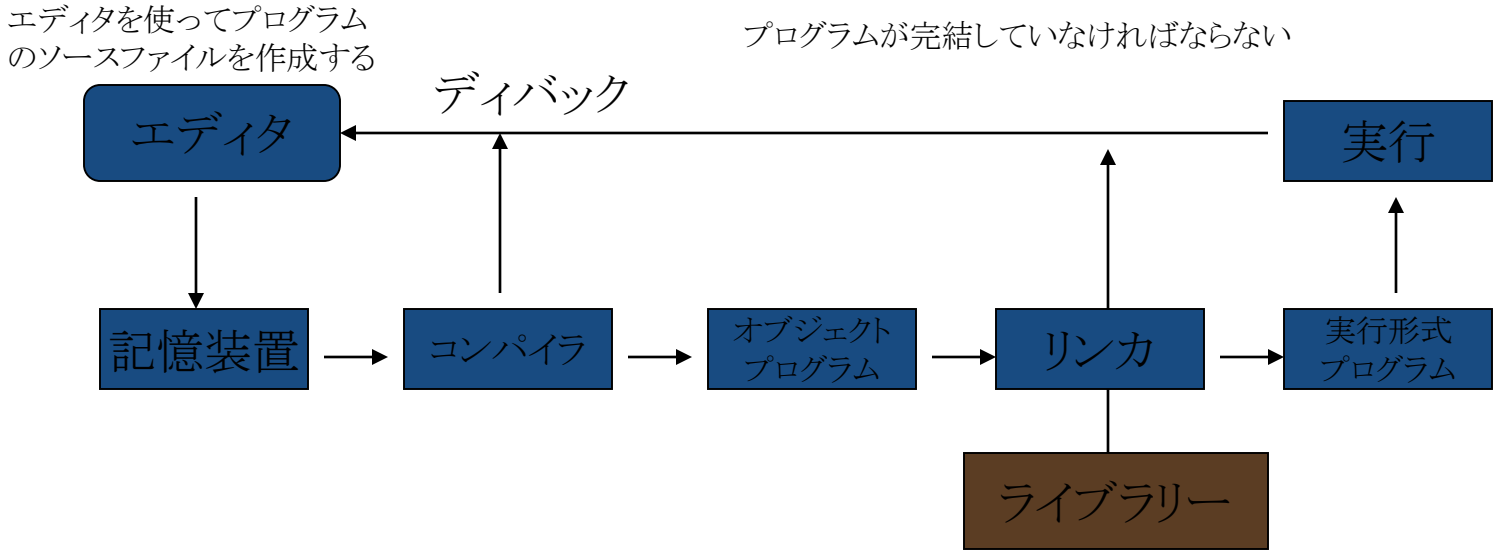
■原始プログラム(ソースプログラム):xxx.cxx

- ♦プリプロセッサ:g++
- ♦コンパイラ:g++

■目的プログラム(オブジェクトプログラム):xxx.o

- ♦リンカ:g++

■実行可能なプログラム(ロードモジュール):a.out



# 4.2 UNIX

# 4.2.1　UNIXの歴史

- 1969年、AT&Tベル研究所で誕生
  - Multiユーザーを実装したOSとしてミニコンPDP-7で作られる
  - WS(ワークスステーション)の主要なOS
    - MS-DOSはUNIXの機能を真似たパソコンのOS(CP/M)の模造品
  - のちにSystem-V系となる。C言語は1973年にUNIXの移植性を高めるために作られた。
- カルフォルニア大学バークレー校で機能拡張
  - 研修者やコンピューターユーザー主導で開発が進む
    - ネットワーク、ファイル構造の実装、現在使われるほとんどのユーティリティーが開発される
  - のちにBSD UNIXの元となる。
- 1980年代 SUNワークスステーションの普及
  - SUN(Stanford University Network)はネットワークワークステーションとして開発

■1990年代:インターネットの発達と共に普及。各社から複数のバージョン

- ♦System-V系 HP-UX, IRIX (Scilicon Graphix)
- ♦BSD系 SUN OS, FreeBSD, NetBSD, OpenBSD
- ♦独立系 Linux, AIX (IBM)

■標準化

◆POSIX (Portable Operating System Interface for Unix) (IEEE P1003)

- ♦IEEE(米国電気電子学会)によって策定された標準インターフェース規格 POSIX.1～POSIX.22に分かれている
  - POSIX.1:ソースレベルの移植性を実現するための標準規格
  - POSIX.2:シェルとツール
  - POSIX.3:POSIX準拠のテストと検証

◆X/Open

- ♦UNIXの標準規格を決めるための業界団体
- ♦UNIXはX/Openが管理する商標

# UNIXの利用先

## ■インターネットの基幹サービス

- ♦DNS (ネームサーバー)
- ♦Web server
  - YahooはFreeBSDをGoogleはLinuxベース独自クラスタリングマシンを利用している
- ♦FTPサーバー
- ♦Mailサーバー
- ♦Firewall　などネットワーク基幹サービス全て

## ■エンジニアリングワークスステーション

- ♦CAD (Computer Associated Design)
  - 車、航空機エンジン、マイクロプロセッサー設計
- ♦Pixar社に代表されるCG作成

## ■先端科学計算

- 大型物理実験
- 大気海洋シミュレーション
- ゲノム解析
- 流体計算
- 多体系問題

# UNIXの機能

## ■計算機資源の管理

### ◆ユーザーの管理

- ♦UNIXは複数ユーザー使用を前提としているためlogin管理などを行う

### ◆アプリケーションプログラムなどのジョブ管理

- ♦UNIXは複数のプログラムが走っている。それらのプログラムをスムーズに走らせるためのメモリーやCPU管理を行う。

### ◆アプリケーションプログラムからの入出力処理の要求の処理

- ♦アプリケーションプログラムがハードウエアに対して要求する入出力に応じてハードウエアを制御する。

### ◆ネットワーク管理

- ♦UNIXのネットワークはEthernet規格やISDNなどの回線による方法でネットワークを構築できる。これらやめーる送受信などを管理する。

### ◆その他ハードウエア、ソフトウエアに関する管理

# 4.2.1　シェル

## ■ユーザーから入力されたコマンドを解釈し、プログラムを起動するアプリケーション

- ◆Loginユーザーに対しシェルプログラムが起動され、以後の端末入力はシェルの標準入力とされる（シェルを通してシステムと対話する）
  - ♦いろいろなシェル
    - sh:　　Bourne shell (UNIXオリジナルのシェル)
    - bash:　　GNUで作られたsh互換のシェル
    - csh/tcsh/zsh　　Berkeleyで開発されたC言語的シェル
    - ksh/pdksh　　高機能なKornシェル
- ◆入力文字列はシェルへの命令として解読され、シェルプログラムの機能の呼び出しもしくは様々なプログラムの起動がなされる。

# シェル：ユーザーから入力されたコマンドを解釈し、プログラムを起動する

アプリケーション

## ■シェルの役割

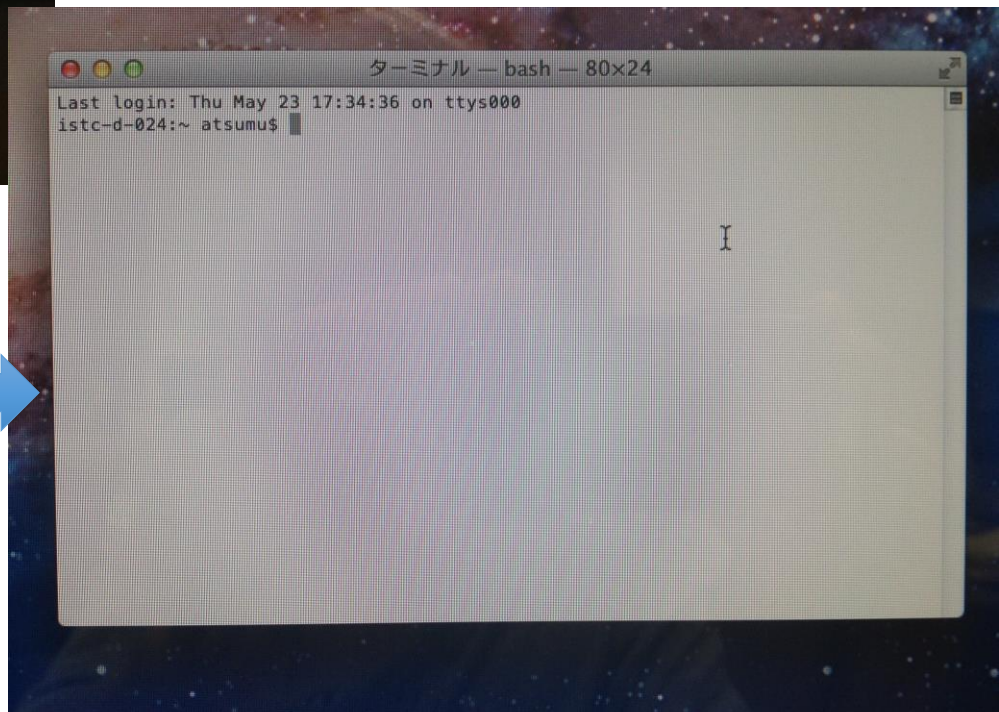
- ◆ユーザー・インターフェース
  - ♦UNIXカーネル* の機能を利用する
- ◆環境設定ツール
  - ♦シェル変数
  - ♦環境変数
    - アプリケーションプログラムやバッチプログラムに対して、ユーザーが設定する変数
- ◆プログラミング言語
  - ♦インタプリタとして働く
    - シェル・スクリプト

カーネル*：UNIXの核となる部分で、マルチタスクやファイルシステム、仮想記憶、入出力など、OS にとって重要な機能を司る。

# 準備：ターミナルの起動

クリックする。

ターミナル上でいろいろな
コマンドを実行する。



# UNIXコマンド(システム関連)

## ■システム関連コマンド

$hostname　　→ ホスト名(計算機に与えられる名前)を表示
$uname -sr　　→ OSが表示される
$which コマンド名　→　コマンドの絶対pathを表示
$who →　ログインしているuserを表示

$history　→　過去のコマンドを表示
$man コマンド名　→コマンドの説明を表示　q　で終了
$echo 引数　　→　引数として与えた文字が表示される
コマンド1 | コマンド2　→ コマンド1の出力をコマンド2の引数とする(パイプ)
コマンド　&　→　コマンド　をバックグラウンドで走らせる

UNIXではコマンドを途中まで打って
TABキーを押すと自動で補完してくれる。

## ■環境変数関連コマンド

$printenv　→　環境変数の表示

$setenv *変数名 値* → *環境変数に値を*代入

$unsetenv*変数名* → *環境変数を*設定解除

## ■重要な環境変数

◆PATH: コマンドを入れているディレクトリ

◆USER: ユーザー名

◆SHELL: シェル名

◆HOST: (現在のシェルを使っている)計算機名

◆DISPLAY: (X端末の)ディスプレイ名

# 演習4-1:UNIXコマンド(システム関連)

$hostname
  ホスト名は何か？
$uname -sr
  OS名は何か?
$who
  誰がログインしている?
$man hostname
  hostnameコマンドの使い方は?
$man man
  manコマンドの使い方が表示される。
$printenv
  SHELL変数はなんだろう。
$echo  $SHELL
  前のコマンドで確認した$ SHELLと一致するか?
  その他の環境変数も確認してみよう。
$echo  SHELL
  前のコマンドとの違いは?

# 4.2.2 ファイルシステム

- ■二次（補助）記憶装置*を効率的に利用するために用意された機構
  - ◆ファイルという概念を導入。物理的実体とは無関係に二次記憶装置上に連続した一塊のデータを記録
    - ♦OSを構成するプログラムやほとんどすべてのI/Oデバイスもファイルとして扱われる。
  - ◆ファイルには名前が付帯される。
  - ◆ディレクトリと呼ばれるファイルの論理的な記憶位置が与えられる。
    - ♦ファイルには所有者や所有グループの概念が与えられ、それに基づきアクセス制限がされる。
    - ♦作成や変更、参照の時刻など、管理上の情報も付加される。

二次（補助）記憶装置*：ハードディスク、光ディスクなどの記憶装置（ストレージ）のこと。⇔一次（主）記憶装置（メモリ）

# ファイルシステム

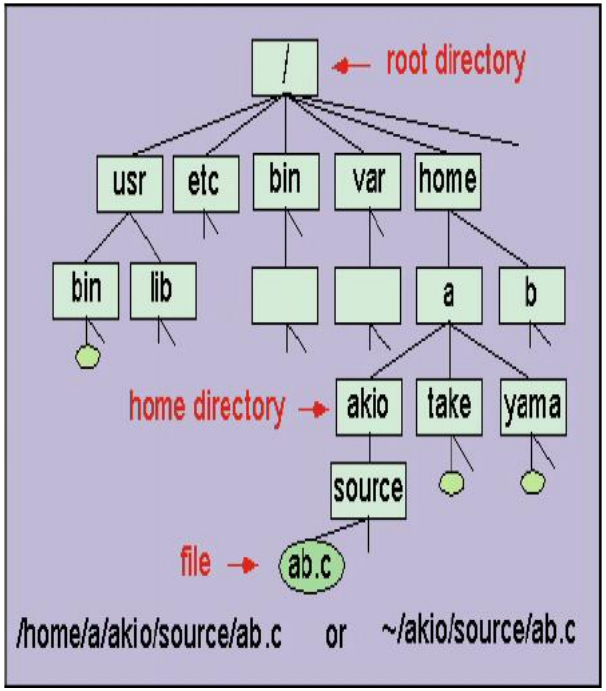
## ■ディレクトリツリー

◆ツリー構造は、rootディレクトリと呼ばれる根にあたる部分を頂点に逆向きの木のような構造である

## ■パス

◆path(パス)とは、目的のディレクトリあるいはファイルまでの道筋を表している。

- ◆絶対パス
  - ◆ルートディレクトリから順を追って目的のファイルを示すパス
  - ◆/home/a/akio/source/ab.c
- ◆相対パス
  - ◆現在いるディレクトリから目的のファイルまでを示すパス
  - ◆./source/ab.c
  - ◆. は現在いるディレクトリを示す
  - ◆.. は一つ上位のディレクトリを示す



# 4．2．3 プロセスとスケジューリング

■プロセス

◆複数のプログラムが見かけ上同時に実行され、それぞれがプロセスとして管理される。

◆プロセスにはその使用者・使用グループが割り当てられる。

■スケジューラー

◆プロセスを起動・中断・再開・終了させる

◆優先順位の変更

♦プロセスに割り当てられた優先順位に従う。

♦CPUを使用し続けると優先順位が下がる。逆に遠のくとあがる。

♦入出力などの起動と完了でプロセスの状態は変わる。

■プロセスの状態

◆実行状態

◆実行可能状態（CPUの割り当て可能状態）

◆待ち状態（入出力、または、その他の待ち状態）

# 演習4-2:ファイル管理コマンド

コマンド集を見ながら以下の操作をしてみよう。

$pwd

何と表示されるか

$pwd

　　パスを確認

$touch *xxxx.yyyy*

*xxxx*: given name　　*yyyy*: familiy name

例:touch kentaro.miuchi

$ls

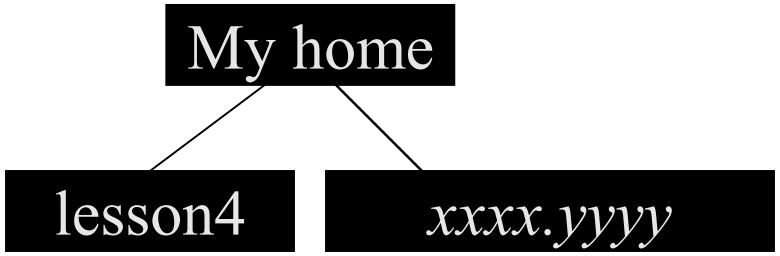
　　ディレクトリの中身を確認

$mkdir lesson4

$cd lesson4

$pwd

　　パスを確認



$cp ../xxxx.yyyy ./

ファイルをコピー

‘.’はカレントディレクトリ、..は一つ上のディレクトリを示す

$cp xxxx.yyyy .temp

$ls

xxxx.yyyyがあるはず。

でも.tempは見えない。

‘.’ から始まるファイル名は不可視ファイル

$ls -a

-a オプションで全てのファイルを表示

$ls -al ~/lesson4

同じものがあるはず

‘~/’は自分のホームディレクトリを指す。

$mv .temp temp1.txt
　ファイルの名前を変更する

$rm temp1.txt
　ファイル消去

$ls -Flと入力してファイル消去を確認

$ man ls > ls.txt
　‘man ls’の内容を’ls.txt’に書き出す
　‘> filename’は出力先を標準出力からfilenameに変更(リダイレクト)する。

$ less ls.txt
　ls.txtの内容を見る　qで終了

$ wc ls.txt

$ file *.*
　ls.txtはどうなっているか？

$ man ls | less

$ cd ..

$pwd
　コマンドでパスを確認

$ file lesson4
　コマンドを入力する 何と出力されるか？
　lesson4/: directory

$ ls -FRl (Lの小文字)
　何と出力されるか？

$ history

# 演習4-3:プロセスの操作

$/usr/X11R6/bin/xload &

$/usr/X11R6/bin/xeyes &

$/usr/X11R6/bin/ico &

$jobs

　　表示を確認

$ps

　　プロセスIDを確認しよう

$ ps aux | grep 14xxxx | less と入力

　　‘ps –aux’で全てのユーザーのプロセスを詳細表示

　　‘grep 14xxxx’でその中から自分のアカウントに関係するもののみ

　　検索を*process_id*調べる

$ kill *process_id*

　　プロセス’ico’ に対応する*process_id*を与えて終了する。

# linux コマンド集①

この他にもコマンドあり。自分で調べて使い倒そう

☆☆☆　必須
☆☆　知ってると便利
☆　慣れたら覚えよう

- C-:「ctrl」を押しながらの意味

## ＜プロセス関連＞

☆☆☆　C-c　ジョブの終了
☆☆☆　jobs　バックグラウンドで実行中、または停止中のジョブを表示。

☆☆　ps　そのユーザーの端末でのプロセスを表示
（ps -au[username]でユーザーの全プロセスを表示）
☆☆　kill %job_id →プロセスを終了する
☆☆　kill *process_id*　→プロセスを終了する
☆☆　&　を付けてコマンド実行　バックグラウンドでジョブ実行
☆☆　C-z　ジョブの停止
☆☆　bg　中断したジョブをバックグラウンドで走らせる

☆☆　fg　中断したジョブをフォアグラウンドで走らせる

## ＜ファイル操作＞

☆☆☆　cp *source destination*　*source*を*destination*へコピーする）
☆☆☆　mv *source destination*　*source*を*destination*へ移動する）
☆☆☆　rm *filename*　ファイルを消す
（rm –i　でファイルを消すかどうか聞いてくるので安全。
rm -R *directory_name*　ディレクトリを中のファイルごと消す。）

☆☆　rmdir *directory_name*　(空の)ディレクトリを消す
☆☆　cat *filename*　ファイルの中身を確認する
☆☆　less *filename*　ファイルの中身を確認する
(スペースで進む　qで終了)
☆wc filename　→ filenameの行数、語数、byte数を表示する
☆　grep *pattern filename*　*filename*の中の*pattern*の含まれる行のみ出力
☆　touch *filename*　*filename*というファイルを作る
☆　file *filename*　ファイルの種類を確認する

## ＜ディレクトリ操作＞

☆☆☆　ls *directory名*　*directory*の内容表示
（引数省略でcurrrect directory）
☆☆☆　pwd current directoryのpath表示

☆☆☆　mkdir *directory名*　*directory*を作成

☆☆☆　cd *directory名*　*directory*に移動

## ＜リダイレクション＞

☆☆☆*Command1*　>　*filename*　*command1*の出力をファイルに書き出す
☆☆　*command1*　>!　*filename*　*command1*の出力をファイルに上書きする
☆☆　*command1*　>>　*filename*　*command1*の出力をファイルに書き足す
☆☆　*command1*　|*command2*　*command1*の出力を引き数としてcommand2を実行する
☆　*command1*　|tee　*filename*　*command1*の結果を出力しながら*filename*に書き出す

# linux コマンド集②

この他にもコマンドあり。自分で調べて使い倒そう

☆☆☆　必須

☆☆　知ってると便利

☆　慣れたら覚えよう

＜ソフト＞

☆☆☆　kterm　ターミナル

☆☆☆　emacs　　editor

☆☆　gnuplot　　グラフ化ソフト

☆☆　vi　　editor (ZZで終了)

# 4.3　C言語の第一歩

# 4.3.1 プログラミング言語

- コンピュータに指示をするために作られた言語
- 低水準言語
  - 計算機が直接理解できる命令・処理と強く関連している言語
  - 小さなプログラム
  - メモリサイズが小さく処理速度が速い
- 高水準言語(高級言語)
  - 人間の言語・思考に近い言語
    - FORTRAN, COBOL, Basic
  - 大規模・複雑なプログラム
  - 処理速度は遅い
  - 計算環境に依存しない(抽象性が高い)

# 4.3.2 C言語

- 1970年代初頭　AT&Tベル研究所のD.M. Ritchieが開発
- UNIXシステム記述言語として利用
  1988年 ANSI*1規格として確立
- 特徴
  手続き型言語
  **記述された命令を逐次的に実行し、処理の結果に応じて変数の内容を変化させていくプログラミング言語**
  構造化言語
  **コンピュータのプログラム上の手続きをいくつかの単位に分け、メインとなる処理では大まかな処理を記述し、サブルーチン*2によって細部を記述していくという方法をとる言語**

*1ANSI（アンスィー、アンジ）：工業的な分野の標準化組織であり公の合意形成のために、さまざまな規格開発をおこなう団体、米国規格協会・米国標準協会 (American National Standards Institute) の略で、アメリカ合衆国の工業的な分野の標準化組織であり公の合意形成のために、さまざまな規格開発をおこなう団体。

*2サブルーチン：プログラミングにおいて、コード中で必要とされる処理を一つのプログラムにまとめて外部から呼び出せるようにしたもの。

# C言語の特徴

- プログラムは関数の集まりである。
  - プログラムの単位を関数と考える。
  - 実行モジュールは必ずmain関数を持つ。
- データ構造が豊富である。
  - 配列の他にもリスト構造等のデータ構造が使える。
- 豊富な制御構文
  - if then … else …, for , case, while 等
- プリプロセッサ処理ができる。
  - 事前にヘッダーファイルの前処理ができる。
- ポインタが使える。
  - メモリアドレスを扱えるので、ハードウエアを直接制御できる。
- 実行性能が高い。
  - 高度な最適化手法
  - 個々のシステムに応じた最適化が可能

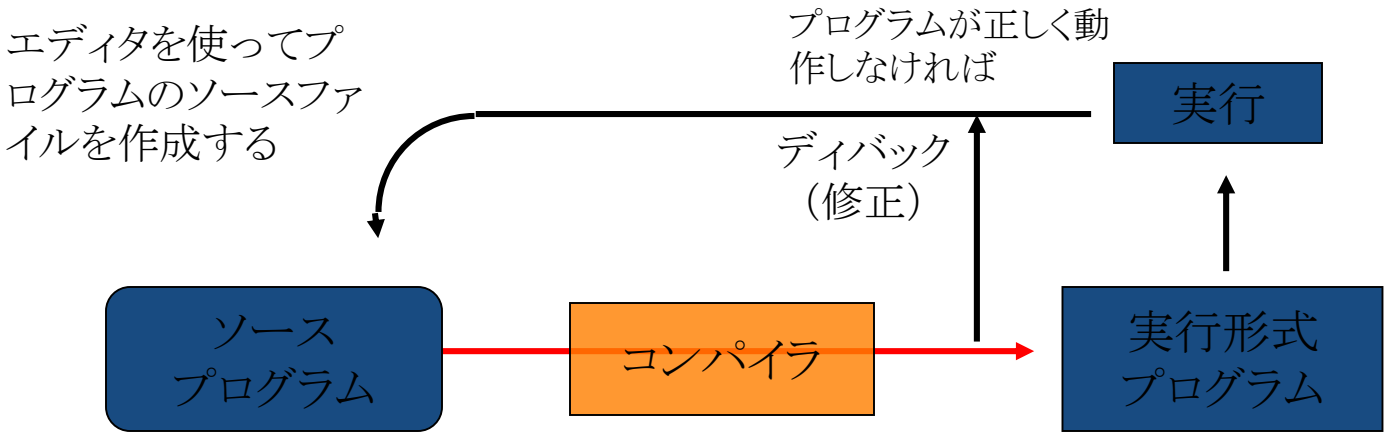
# プログラム作成から実行まで

実際に行う手順

- ソースプログラム作成：xxx.cxx

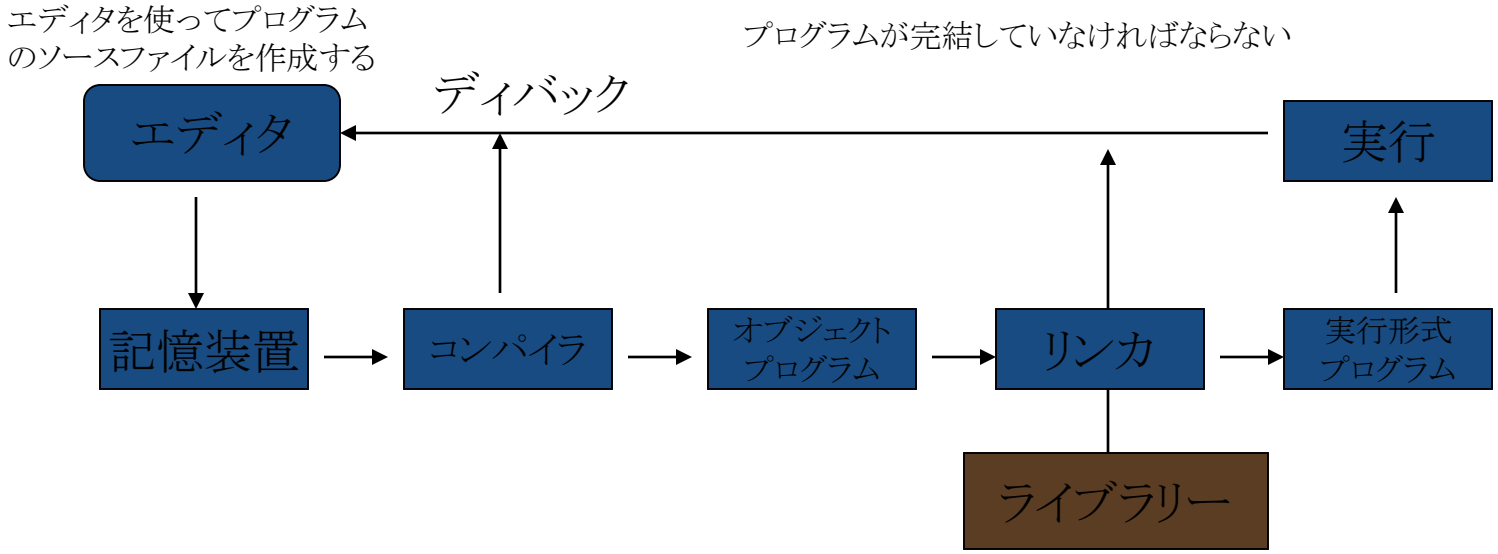
↓ コンパイラ：g++

- 実行可能なプログラム：a.out



# プログラム作成から実行まで

- ソースプログラム：xxx.cxx
  - プリプロセッサ：g++
  - コンパイラ：g++
- オブジェクトプログラム：xxx.o
  - リンカ：g++
- 実行可能なプログラム（ロードモジュール）：a.out



# 演習4-4：C言語のコンパイルと実行

全体の流れ

1. 本日のページから hello.cxx というソースファイルを拾ってくる。
2. 該当directoryに移動
3. g++コンパイラでhello.cxxソースからhelloという実行ファイルを作る。
4. 作成した hello プログラムの実行

# 演習4-4:C言語のコンパイルと実行

ブラウザで本日のページを開き、hello.cxxを右クリック「リンク先のファイルを別名で保存」する。拡張子txtはつけない。場所は各自のアカウント名を選択する。

$cd ~/lesson4
　　lesson4に移動

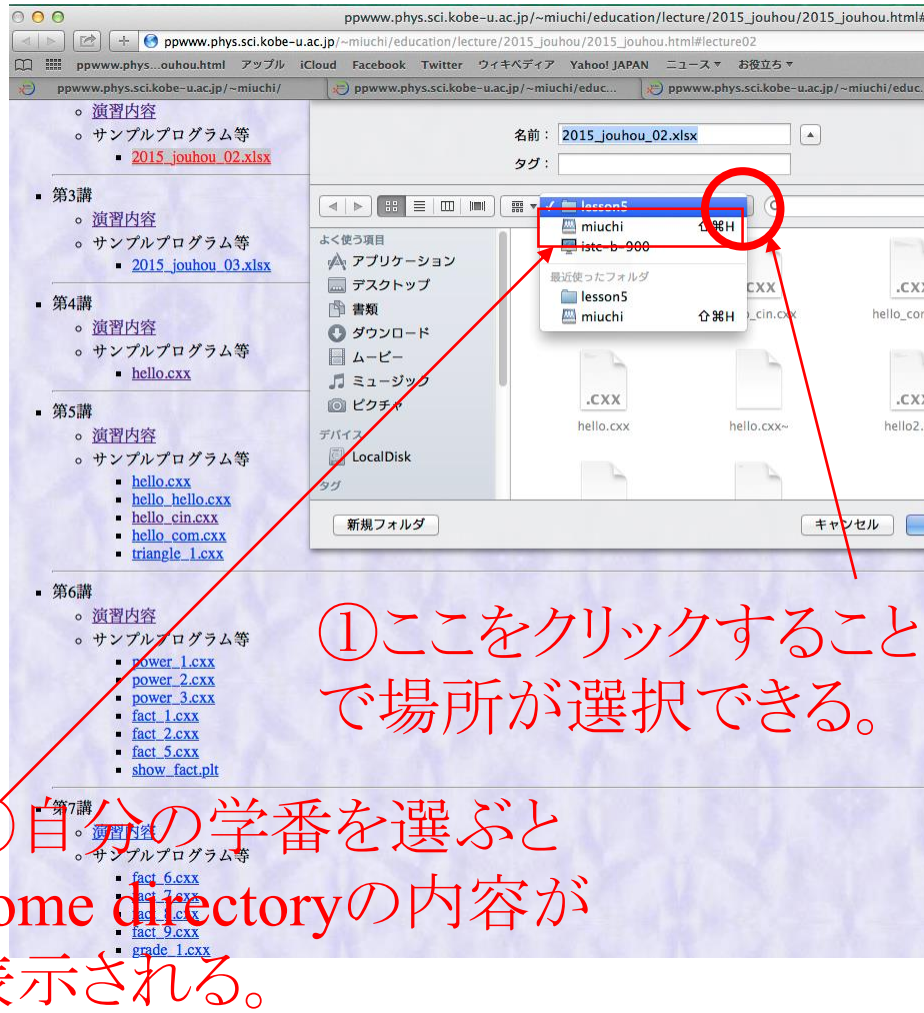
$cp ../hello.cxx ./
　　コピー

$ls
　　hello.cxxが存在することを確認

$g++ -o hello hello.cxx
　　コンパイル

$ls
　　helloが存在することを確認

$./hello



file名 : 2015_jouhou_04_学籍番号の下4桁.txt

# 課題4-1:UNIXコマンド、C言語第一歩

> >>などのリダイレクションを用いて、上記の指定ファイルに
host名
shell名
~/lesson4 でのls -l の実行結果
~/lesson4 でのwc hello.cxx の実行結果
~/lesson4 での./hello の実行結果

を記録せよ。
提出前に fileの中身を確認すること。

# 課題提出

- 宛先　fsci-phys-jouhou@edu.kobe-u.ac.jp
- 件名　2015-report04_**学籍番号**の下4桁
- 本文　学籍番号と名前
- 添付ファイル：2015_jouhou_04_学籍番号の下4桁.txt
  - 提出前にfileの内容を確認。
- 締め切り　2014年5月8日（金）13：00