

第2章 オペレーティングシステムの ユーザインタフェース

花田 英輔

(このPowerPointは渡辺名誉教授作成のものを花田が一部改題した)

1

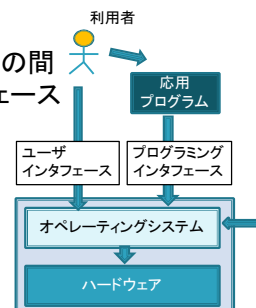
OSのインタフェース

▶ ユーザ と OS の間

⇒ユーザインタフェース

▶ 応用プログラム と OS の間

⇒プログラミングインタフェース



2

OSの利用者(教科書2.1)

▶ 一般利用者

- 通常の利用
- コンピュータ利用に詳しくない者にも分かりやすく使いやすく

▶ システム管理者

- 呼称: アドミニストレータ(Windows)、ルートユーザ(またはスーパーユーザ)(UNIX)
- システムの管理
- システム全体の変更に関わることを実行
- 特別な権限付与、特別な命令実行可能

3

一般利用者とシステム管理者

▶ なぜ分離するのか？

- 多数の利用者が各々勝手に設定をいじったらまずい
- 個人利用でも管理者権限＝通常は一般の権限で利用
危険な操作は特別な権限を得て可能とする

▶ UNIXでの習慣

- 管理作業時も、最初のログインは一般ユーザで入り、その後に管理者へ移行⇒危険を最小化、複数管理時に作業者の記録

▶ Windowsでは？

- 以前: パーソナルユースのため「ユーザ」の概念無し
- 現在: 管理者と一般ユーザは分離可能
 - ・「Windowsへのログイン」が可能になったため

4

ユーザインタフェース

▶ 利用者がOSを操作するインタフェース

▶ GUI(Graphical User Interface)

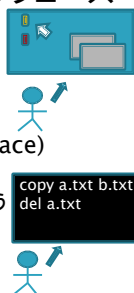
- アイコン、マウスを使った操作

▶ CUI(Character-based User Interface)

- コマンド入力による操作
- CLI(Command Line Interface)とも言ふ

▶ 他のUI(User Interface)も提案

- 音声、ジェスチャなど

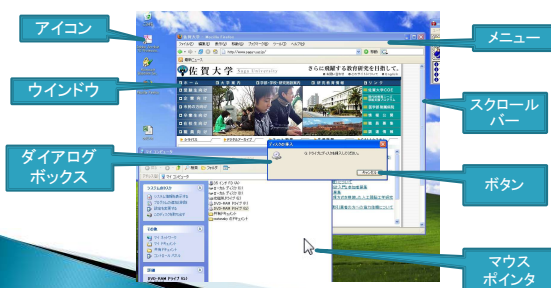


5

グラフィカルユーザインタフェース (教科書2.2)

▶ GUI、Graphical User Interface

▶ アイコン表示、マウス操作



6

GUIの歴史

- ▶ 1974年、Xerox社PaloAlto研究所のAlto
 - ビットマップディスプレイ=任意位置に任意の図形表示
 - マウス=画面上の位置を指すための装置
- ▶ Macintoshに採用されて普及



7

ビットマップ・ディスプレイ

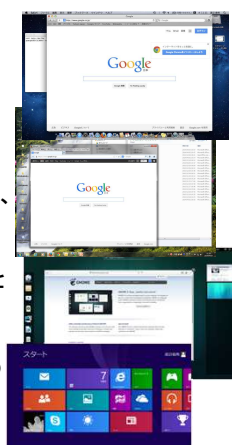
- ▶ ビットマップ・ディスプレイ
 - 現在ではほとんどすべてが、この形式
 - 図や文字をドット(ピクセル、画素)に分解して、画面のドット単位に表示・非表示制御
- ▶ キャラクタ・ディスプレイ
 - 文字コードを受け取り該当文字列を表示
 - Windowsのコマンドプロンプトのような表示



8

主なGUI

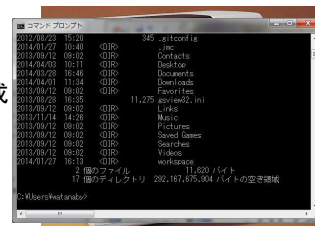
- ▶ MacOSのGUI
 - Macintoshの画面
- ▶ WindowsのGUI
 - MacOSに続き採用
 - CUI基本のMSDOSに機能追加、後にGUI基本のOSに
- ▶ X Window System
 - UNIXの標準的のウィンドウ機能として普及
- ▶ タッチベースのUIへ
 - iOS、Android、Windows(8以降)
 - GUIの一種ではあるが、既存GUIからの変更点多い



9

CUI(教科書2.3)

- ▶ CUI: Character-based User Interface
- ▶ CLI: Command Line Interface
- ▶ コマンド
 - OS操作のため、利用者から与える指示文
 - 短い単語や略語で構成
- ▶ コマンド言語
 - コマンドの体系
- ▶ UNIXコマンド
 - UNIXで使用
- ▶ MSDOSコマンド
 - Windowsの先祖であるMSDOSで使用
 - Windowsもコマンドプロンプト上で使用



10

主要なUNIXコマンド(p.27)

- ▶ cd .. 作業ディレクトリの移動
- ▶ ls .. ディレクトリ内のファイル一覧表示
- ▶ cp .. ファイルコピー
- ▶ rm .. ファイル削除
- ▶ mkdir .. ディレクトリ作成
- ▶ rmdir .. ディレクトリ削除
- ▶ more .. ファイルを表示
- ▶ pwd 作業ディレクトリ名を表示
- ▶ ps .. プロセス状態を表示
- ▶ date 日付、時刻を表示
- ▶ man .. マニュアルページ表示
- ▶ grep .. ファイルからパターンを探す
- ▶ gcc .. Cコンパイルする
- ▶ vi .. 編集する
- その他多数。自分で作成することも簡単。
- プログラミング演習で作成のコンソールアプリ

使用例

```
$ ls
bin      progl.csv
test.c   test2.c

$ more test.c
#include <stdio.h>
int main(void){
    fputs("Hello World\n", stderr);
    return 0;
}

$ gcc test1.c -o test1
$ ./test1
Hello World
$
```

ファイル一覧表示
ファイル内容表示
コンパイル
実行
次のコマンド待ち

11

UNIXのシェル(p.27~28)

- ▶ シェル
 - UNIXでユーザインタフェースを提供するソフトウェア
 - コマンドを受け付け、解釈してプログラムを実行
 - OSの外郭を担うので、Shell(貝殻)と呼ばれる
- ▶ 便利機能
 - リダイレクション
 - データの入出力をファイルへ変更
 - パイプ
 - データを別のコマンドへ受け渡し
 - シェルスクリプト
 - 複数コマンドを記述したファイルを実行、IFなども可

```
$ ls
bin      progl.csv
test.c   test2.c

$ more test1.c
#include <stdio.h>
int main(void){
    fputs("Hello World\n", stderr);
    return 0;
}

$ cc test1.c -o test1
$ ./test1
Hello World
$
```

12

リダイレクション

- 標準入出力を使うコマンドに対して
 - コマンド < 入力ファイル名 : 指定ファイルから読み込み
 - コマンド > 出力ファイル名 : 指定ファイルへ書き出し

使用例

```
$ ls -l      ←普通にファイル一覧を表示(画面へ)
bin        progll.csv test.c   test2.c

$ ls > x.txt ←出力リダイレクト(ファイルx.txtへ)

$
```



13

パイプ

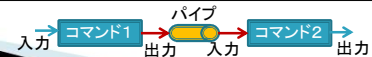
- 2つのコマンドをつなぐ
 - コマンド1 | コマンド2 : コマンド1出力をコマンド2入力へ

使用例

```
$ ls | wc -l  ←ファイル一覧出力を、行計数へ
10           ←1行1ファイル表示のため、ここにファイルは10個

$ ls | grep "txt" ←ファイル一覧出力を文字列探索へ
test.txt      ←ファイル名に"txt"を含むファイル一覧
aa.txt

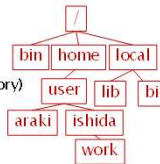
$ ls | grep "txt" | wc -l ←ファイル名に"txt"を含む
                  ファイル数
```



14

UNIXコマンドの使用例

```
cd ..      カレント(現在)ディレクトリを一つ上位(..)へ移動(change directory)
ls         そのディレクトリ内の一覧を見る(list)
cd /       カレントディレクトリをルート(根)ディレクトリへ移動する
cd home    カレントディレクトリを現在のディレクトリからの相対位置指定
           /を最初に付けないと、カレントディレクトリからの相対位置指定
cd /home/user ホームディレクトリ(ログイン時のdefault)に戻る
mkdir work カレントディレクトリの下にディレクトリworkを作る(make directory)
cp t.c work ファイルt.cを、ディレクトリworkへコピーする(copy)
cd work    ディレクトリworkへ移動する
ls ..      一つ上のディレクトリのファイル一覧を見る
rm t.c     ファイルt.cを削除する(remove)
cd ..      一つ上に移動する
rmdir work ディレクトリworkを削除する(remove directory)
```



15

UNIXコマンドの使用例

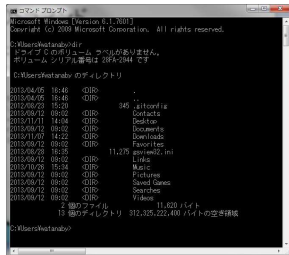
```
ls *.c      Cファイル群の一覧表示
cp *.c progdir Cファイル群をprogdirにコピー
grep "pattern" *.c Cファイル群からpatternを探す
gcc test1.c -o test1 test1.cをコンパイル
./test1     コンパイル結果のtest1を実行
vi test1.c  test1.cを編集ソフトviで編集
diff test1.c test2.c ファイルの差異表示
ls te*.c    te[任意文字列].cのファイル一覧
cat te*.c te_all 上記ファイル群をつなげてte_allに
tar czvf te.tar.gz te*.c 上記ファイル群を1ファイルに圧縮
```

16

MSDOSコマンド

- UNIXに倣うが、より分かりやすい単語使用
- Windowsでもコマンドプロンプトで使用可能

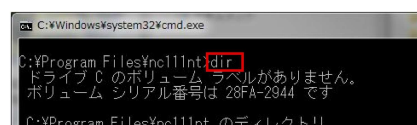
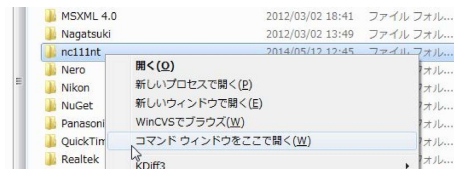
```
dir .. : ディレクトリ内のファイル一覧
cd ..  : 作業ディレクトリの移動
copy .. : ファイルのコピー
type .. : ファイル内容の表示
del .. : ファイルの削除
```



17

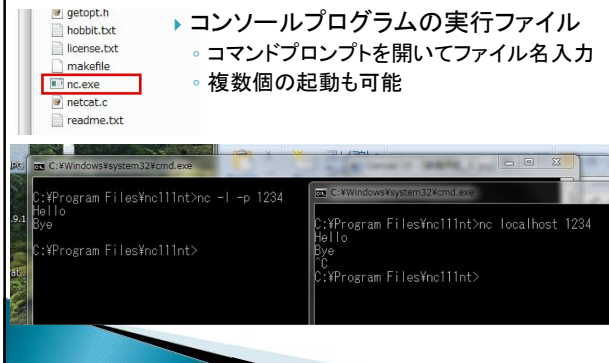
Windowsにおけるコマンド入力

- コマンドプロンプトの起動
 - フォルダを指して「Shiftキー+マウス右クリック」



18

コンソールプログラムの実行



19

CUIの難点と利点

- ▶ 難点: コマンドを覚える必要
- ▶ 利点: 多数の処理を一括して行うときに便利
 - 例:
 - img-連番.jpgをimage2013_連番.jpgに変更
 - 全ての.auxと.logを削除
 - 全てのC++ファイルの中から指定文字列を探す
 - testを含む名前のC++ファイルのみを別の場所に移す
 - 多くのC++ファイルをコンパイル実行する
 - 一定条件を満たすファイル群に対して、同一の処理を行う

20

今回の課題

1. GUIとCUIとは何か。それぞれ説明し、その優劣を比較せよ。
 2. UNIXではすべてをGUIとすることは難しい。コマンド言語でないとできない仕事の例を挙げよ
 3. (予習) Windowsで用いられるDLL(識別子が.dllとなるファイル)とは何か調べて記せ
- ▶ 今回のファイル名は“学籍番号-OS03.docx”
(例: 24238000-OS03.docx)としてください
 - ▶ 締切: 10月24日(金) 18:00 (遅れた場合は減点)

21

課題についての注意事項

- ▶ レポートはWordでA4 2ページ程度(表紙含まず)にまとめること
 - 長さ制限は無いですが、極端に短いものは減点
 - 提出はeラーニングシステムを通じて行うこと
- ▶ 締切はその週の金曜日18:00
 - 11月6日と1月14日は別に指定
 - 提出が遅れた場合は減点
 - ・ 大幅な場合はさらに減点の可能性有

記載時の注意事項

- ▶ 参考資料(Webページ)がある場合は**出典を書くこと**
出典を書かずに引用した場合は減点対象です

22