情報処理 (9)

オペレーティングシステムとファイルシステム

増原 英彦

1　オペレーティングシステム

今日の計算機にとって、**オペレーティングシステム (OS)** は不可欠の存在である。「MS-Windows」「Mac OS X」や「Linux」といった名前のソフトウェアが OS を指していることは、多くの人が知っているであろう。

その一方で、OS が何であるかを説明することは難しい。しばしば「見た目」や「一緒に付いてくるアプリケーションソフトウェア」こそが OS の本質であると誤解した議論がなされていることもある。

ここではまず OS が存在する目的を先に説明し、それから実際に OS がどのような機能を提供しているかを説明する。

1.1　意義

計算機のプログラムは、

- CPU(中央処理 装置) によって実行される。言い換えれば、CPU を 使って 動作する。
- キーボードやスキャナ、ネットワーク装置などの入力 装置 から処理すべきデータを受け取る。
- メモリ (主記憶 装置) に蓄えたデータを使って処理をする。言い換えれば、メモリを 使って 記憶する。
- ディスプレイ装置やプリンタ、ネットワーク装置などの出力 装置 に処理したデータを出力する。

これらの動作をよく見ると、プログラムは計算機内外にある様々な「装置を使って」処理を進めていると言える。

計算機が発明された当初は、プログラムはこれらの装置を**直接**使っていた。しかし、計算機の発展とともに、装置を直接使うソフトウェアを作成することはは次のような理由から難しくなってきた。

一つの理由は、1 つの計算機を複数の人間が使うようになったり、1 人の人間が 1 つの計算機を専有している場合でも複数のアプリケーションプログラムを実行するようになってきたことである。もし、装置を直接使うプログラムが同時に実行されていると、2 つのプログラムが同じ装置を同時に使ってしまって混乱する危険がある。例えば、2 つのプログラムが同時にプリンタ装置を使ったとしたら、両方の出力が 1 枚の紙に混在してしまうかも知れない。従って、複数のプログラムが同時に実行されたとしても、装置類が混乱なく使えるように、調停をする必要がある。

もう一つの理由は、計算機ハードウェアの急速に進歩である。計算機が発明された当初は、1 台 1 台の計算機ごとに違ったプログラムを作っていた。その場合、プログラムは、その計算機に接続されている装置—例えばディスプレイ—を直接使っていたとしても問題はなかった。しかし、計算機に接続

されているハードディスクやネットワークインターフェースなどのハードウェアは、日々改良されるようになった現在では、同じプログラムを異なる装置の接続された計算機で実行できるべきである。そのため、アプリケーションプログラムが装置類の違いを気にすることなく装置を使えるような仕組みが必要である。

1.2 基本的な役割

上の 2 つの理由から、現代の計算機は**オペレーティングシステム (以下 OS)** と呼ばれるソフトウェアを**アプリケーションプログラム**とともに動作させ、アプリケーションプログラムは OS を通して計算機内外の装置を利用するようになっている。別の言い方をすると、OS はアプリケーションプログラムのために CPU・メモリ・入出力装置などを管理するソフトウェアであると言える。

OS の働きは次の 2 点に要約できる:

- 複数のアプリケーションプログラムが同じ装置を使っても問題が起きないように調整をする。例えば、以下のようなことである。
  - アプリケーションプログラムが使うメモリを決める
  - 1 つのアプリケーションプログラムが一定時間動いたら、一時停止させ、別のアプリケーションプログラムを動かす—これによって見掛け上複数のアプリケーションプログラムが「同時に」動く
  - 1 つアプリケーションプログラムがプリンタに印刷内容を送っているときに、他のアプリケーションプログラムがプリンタにデータを送ろうとしたら印刷が終るまで待たせる
- アプリケーションプログラムが、実際に計算機に接続されている装置の詳細を知らなくとも装置が使えるように処理を代行する。例えば、
  - マウスボタンが押されたかどうかを調べるには、計算機ごと (マウス装置の種類ごと) に違った調べ方をしなければいけない。そこで、アプリケーションプログラムは OS に「マウスボタンが押されたか」を問い合わせるようにする。そうすると、OS は現在接続されているマウス装置に応じた調べ方をして、その結果をアプリケーションプログラムに通知する。

  このようなことを可能にするために、OS には接続されている装置を扱うプログラムが装置のハードウェアごとに用意されている。そのようなプログラムは **デバイスドライバ**と呼ばれる。

1.3 働き

計算機が動く際には、ハードウェア、OS、アプリケーションソフトウェアの 3 つがそれぞれ動いて一つの処理を完了することになる。以下では、具体的な動作について、これらがどのように役割を分担しているかを説明する。

アプリケーションプログラムの起動: すでに説明したようにプログラムとは CPU に対する命令の並びであり、メモリに格納されている。アプリケーションプログラム (電子メールクライアントや WWW

ブラウザなど) は、**実行中のものだけが**メモリ上に格納されている。これは、計算機のメモリは限られた量のデータしか格納できないためである。

そのため、アプリケーションプログラム (の命令の並び) は、普段はファイルに保存されている。アプリケーションが**起動される**とき、OS はファイル中の命令の並びをメモリに読み込み、実行を始める。その後は、CPU がそのアプリケーションプログラムの命令を実行することになる。(人間がアプリケーションを起動するときには、後述する**シェル**を通して OS に指示を出すことで起動している。)

練習 9–1: (アプリケーションプログラムの実行) 以下のようにしてファイルの中身を表示するアプリケーションプログラムを実行してみよ。

1. 「Emacs」あるいは「テキストエディット (TextEdit)」を使い、適当な名前のファイルを作り、適当な内容を書く。ここでは「Documents」フォルダの中に「memo.text」というファイルを作ったとする。

2. Mac OS X の「ターミナル (Terminal)」を起動し、

   `cat Documents/memo.text` Enter

   と入力する。この指令は「Documents」というフォルダの中にある「memo.text」という名前のファイルの中身を表示せよ (cat[1]) という意味である。

練習 9–2: (アプリケーションプログラムの実態) 上の 練習 9–1 では、cat というアプリケーションプログラムを使用した。このプログラム自体は、「bin」という名前のフォルダの中にしまわれている。そこで、Mac OS X の「ターミナル (Terminal)」を起動し、

`cat /bin/ls` Enter

と入力すれば、その中身を見ることができる。どのような内容が表示されるか? (なお、/bin/ls のように先頭に「/」が付いているが、これは**ルートディレクトリ**の中にある「bin」というフォルダという意味になる。詳しくは HWB の「13. ファイルシステム」を参照せよ。)

なお、文字化けした場合は、一度「ターミナル」ウインドウを閉じて、新しいウインドウを開くとよい。

複数のアプリケーションの同時実行: 1 つの計算機には 1 つの CPU しかないことが多いにも関らず、複数のアプリケーションが同時に動いている。これは、(1) 計算機内部の時計が定期的に OS のあるプログラムを実行しているハードウェア上の仕掛けがあり、(2) そのプログラムが、現在実行されているアプリケーションを止め (どの命令まで実行したかを覚えておく)、(3) 別のアプリケーションを再開する (覚えておいた命令の次から実行する)、ことで実現されている。この切替は人間とっては充分に短い時間間隔で行われているので、見かけ上、複数のアプリケーションが同時に実行されているように見える。

[1]何故「中身の表示」が “cat” という名前になっているかについては、理由があるのだが、ここでは気にしないことにする。

練習 9–3: (同時実行) 次のようにして、複数のアプリケーションプログラムが同時に実行されていることを確めよ。

1. Mac OS X の「ターミナル (Terminal)」を起動し、

   `top` Enter

   と入力すると、その計算機で実行されているプログラムに関する情報が一覧表示される。画面の下側の「COMMAND」と書かれた列がプログラムの名前である。(その他に、CPU の時間やメモリをどのくらい使っているかといった情報が表示されている。興味がある場合は、別の「ターミナル」ウインドウから `man top` Enter としてマニュアルを参照せよ。)

2. アプリケーションプログラム (「Mail」や「Safari」など) を起動したり終了みよ。上の表示がどのように変化するか観察せよ。なお、この`top` コマンドは、アプリケーションプログラムだけでなく、OS が「こっそり」実行しているプログラムも表示している。

3. この`top` コマンドを終了するには「q」をタイプする。

メモリの管理: アプリケーションはメモリ上に置かれたデータを読み書きすることで情報を処理する。このとき、同時に実行されている別のプログラム (アプリケーションや OS) のプログラムやデータが置かれているメモリを使ってしまうとおかしなことになる。

このようなことが起きないように、アプリケーションが使うメモリは OS が割り当てて使っている。OS によってはハードウェアの機能を活用して、別プログラムのメモリを読み書きできないように制限している場合もある[2]。

入出力装置の管理: 例えばハードディスク装置にデータを記録する場合、CPU は「x 枚目の円盤[3]の y 周目の先頭から z バイト目から w バイト分次のデータを書け」といった指示を出している。しかし、アプリケーションがデータを保存する際にも上のような数による指示を直接出すのは次のような理由から不便である:

- 新しく保存する際に「どこが空いているか」を知るのが難しい
- 結局そのアプリケーションを使う人間が数を直接指定することになる
- 装置を変えたときに数が変わってしまうかも知れない

そこで、アプリケーションは OS を通して間接的に入出力装置を使うことにする。このときアプリケーションが操作するのは、ハードウェアによる違いが分からないような抽象化された入出力装置であり、これを**ファイルシステム**(後述) という。

[2]Unix, MS-Windows NT/2000, MacOS X などがそうである。他のアプリケーションに割り当てたメモリを読むこともできないので、1 つの計算機を複数の人が同時にメールリーダを使っていたとしても、メールを盗み読みされる必配はない。逆に MS-Windows 95/98, MacOS などはそうしていないので、プログラムの誤りによって計算機システムが「不安定」になりやすい。

[3]雑な例えをすると、ハードディスク装置は磁気的に記録のできるレコード盤を何枚か重ねたようなものである。

2 ファイルシステム

アプリケーションが OS を経由して入出力装置にデータを読み書きする際には、**ファイル**という単位で指示を出す。具体的には、次のようなことを行っている:

- アプリケーションがある名前でファイルを作れという指示を出す。OS は入出力装置の使われないない場所を探し、その場所を「使用済」にする。同時に、名前と場所の対応を覚えておく。
- アプリケーションがある名前のファイルに書き込みをする/ から読み込むという指示を出す。OS は名前と場所の対応を調べ、入出力装置の適切な場所にデータを書き込む/から読み込む。
- アプリケーションがある名前のファイルを消すという指示を出す。OS は名前と場所の対応を調べ、その場所を「未使用」にする。同時に、名前と場所の対応を消す。

このようにファイルを単位として操作することで、アプリケーションは入出力装置の詳細を知らずにデータの読み書きができる。別の言い方をすれば、入出力装置による違いは OS が隠してしまっているということである。近年の OS では、ハードディスク装置に限らず、ネットワーク装置・キーボード・マウスといった装置類もファイルとして扱われている[4]。また、複数のアプリケーションが同時に読み書きをしようとした場合でも、OS の内部で順に処理されるので、混乱しない。

アプリケーションを使う人間にとっては、沢山のファイルに全て異なる名前をつけるのは容易でない。そこでドメイン名と同様に、グループ化して名前をつける方法がとる。このグループのことを**ディレクトリ**という。当然、ファイルに対する操作と同様にディレクトリに対する操作も OS を経由して行う。

また、複数人が同じ計算機を使う場合には、他人にファイルを読み書きされたくない場合もある。そのため、各ファイルの**所有者**を決めておいて、アプリケーションを実行したのがファイルの所有者かどうかによって読み書きができるかどうか (**パーミッション**という) を制御することが行われている。

練習 9–4: (ファイルの表示) Mac OS X の「ターミナル (Terminal)」を起動し、

`ls -l` Enter

と入力すると**カレントディレクトリ**のファイルの一覧が表示される。「ファインダ (Finder)」の表示と照らしあわせて、どのフォルダの情報が表示されているか、また、ファイル名以外の情報が何であるかを推定せよ。

練習 9–5: (入力装置とファイル) Mac OS X の「ターミナル (Terminal)」を起動し、

`ls /dev` Enter

と入力するとファイルとして扱われている入出力装置の一覧を見ることができる。ただし、それらのファイル対して普通のユーザができる操作はほとんどない。

[4]いわゆる「PC」あるいは「IBM PC/AT 互換機」といわれる計算機では非常に多種の入出力装置類が売られている。アプリケーションがこれらの装置類を使う際には OS を経由して使うため、交換したとしても同じようにして使える。そのために、これらの装置を比較的容易に交換することができるのである。一方、ゲーム機などの場合、接続される入出力装置は限定されているので、アプケーションがこれらの装置を直接操作することも行われている。

3 シェル

上に述べたようにOSには様々な機能があるが、これらはアプリケーションから使うものである。そこで、計算機システムには、人間の指示をOSへ伝える**シェル**[5]というプログラムが用意されている。シェルによって使える機能としては、アプリケーションの実行開始や停止、ファイルの複製・消去や名前の変更、ウィンドウの管理、計算機の設定管理などである。シェルには大きく分けて2つの種類のものがある:

コマンド入力型: キーボードから文字列で指示を入力するもの。指示が文字列になっているので、きめ細かな指示を組み合わせたり、自動化するのに向いている (練習 9–?? 参照)。また、この種のシェルは、ごく基本的な機能だけを用意し、多くの機能をアプリケーションプログラムに任せていることが多い。例: Mac OS X の「ターミナル (Terminal)」で動いている「`bash`」、MS-Windows の「MS-DOS プロンプト」や「コマンドプロンプト」。

直接操作型: マウスでファイルを動かすなどの操作によって指示をするもの。直感的なので、1つ1つ手順を確認しながら処理をするのに適している。一方で、複雑な操作を行う場合や、処理を自動化するのには向いていない。また、この種のシェルは、非常に沢山の機能を持つ傾向にある。例: MS-Windows の「エクスプローラ」や Mac OS の「ファインダー (Finder)」。

4 演習: コマンドとファイルシステム

練習 9–6: (コマンド) HWB「12. コマンド」を読み、コマンドの基本的な使い方を理解せよ。コマンドの基本的な使い方は、次回以降でプログラミングの演習をする際に必須となることに注意せよ。

その際、以下の点に注意せよ:

- 「`ca00000$ cal` Enter」のような入力例は、**「`cal` Enter」の部分だけ**を入力する意味。
- 以下のキー操作によってターミナルウィンドウへの入力を修正することができる:

  左へ1文字移動: Control `b`　　右へ1文字移動: Control `f`
  以前に入力した行を復元: Control `p`　　その逆: Control `n`

- 多くのコマンドは正しく実行された場合には何も表示せず、失敗したときのみメッセージを表示する。実行した後にどのようなメッセージが表示されるか、注意深く読むこと。
- 空白はコマンドとパラメータの区切りとして重要な意味を持っている。HWB では「`cal` Space `1999` Enter」と「`cal 1999` Enter」のように書いている場合があるので、注意が必要である。
- ターミナルウィンドウの動作がおかしくなったときは、次のキーを入力してみるとよい:

  入力の完了: Enter　　コマンドを中止: Control `c`　　表示の一時停止を解除: Control `q`
  入力の終了: Control `d`　　コマンドを強制終了: Control `\`

[5] OS という「核」に人間が扱えるような「殻」(shell) をかぶせているというアナロジー。

練習 9–7: (プリンタの使用) 次の手順に従ってプリンタを使ってみよ。

1. HWB 「20.10.1 教育用計算機端末におけるプリンタの使い方」を読み、プリンタの使い方の概略を理解せよ。
2. 印刷したい内容を用意する。エディタを使って、適当な名前の (試えば`printer.txt`) というファイルを作り印刷したい文章を書け。以下の点に注意せよ:
   - 長い行は上手く印刷できないので 70 字程度で改行せよ
   - 複数の行にわたった、日本字を含む文章で練習しておくとよい
3. HWB 「20.10.2.2 テキストファイルの印刷」に従って上で作ったテキストファイルを印刷してみよ。

練習 9–8: (ファイルシステム) HWB の「13. ファイルシステム」を読み、ファイル・パス名・ディレクトリの概念と基本的な操作を以下に整理せよ:

ディレクトリ (フォルダ): とは何か?

ルートディレクトリ: とは何か? / を表わすパス名は?

ホームディレクトリ: とは何か? / へ移動するコマンドは? / を表わすパス名は? / g999999 という利用者のホームディレクトリを表わすパス名は?

カレントディレクトリ: とは何か? / を表わすパス名は? / を表示するコマンドは? / を変更するコマンドは?

親ディレクトリ: とは何か? / を表わす名前は?

サブディレクトリ: とは何か?

操作: ディレクトリ を作るコマンドは? / を丸ごと復製するコマンドは? / を削除するコマンドは? / の名前を変更するには?

ファイル: の内容を表示するコマンドは? / の一覧を表示するコマンドは? (ファイルの種類を表示するオプションは? / ファイルの詳細情報を表示するオプションは?) / を復製するコマンドは? / を移動するコマンドは? ファイル名を変更するコマンドは? / を削除するコマンドは? / の大きさを調べる方法は?

ファイル名・パス名・ワイルドカード: ファイル名の拡張子とは何か? / 絶対パスとは何か? (例も挙げよ) / 相対パスとは何か? (例も挙げよ) / 以下の条件を満たすようなワイルドカードを使ったパス名 (それぞれ): (i) a, b, c または d で始まる (ii) チルダ (~) で終る (iii) 2 文字目が a か e で、3 文字以上の長さがある (iv) 数字を含む / あるパス名がファイルかディレクトリかを調べる方法は?

ファイルのパーミッション: を表示するためのコマンドは? / が「`-rw-r--r--`」だった場合の意味は? / が「`-rw-rw-rw-`」だった場合の意味は? / が「`drwx------`」だった場合の意味は? / が「`drwxr-xr-x`」だった場合の意味は? / テキストファイルを自分と同じグループの人のみが読めるようにするコマンドは? / テキストをファイルを誰でも読み書きできるようにパーミッションを変更するコマンドは?

5 課題

課題 9–1 および課題 9–2 のレポートをテキストファイルで作成し、それをプリンタで印刷したものを提出せよ。ただし、

- 課題 9–1 を必須とする。余力のある者は課題 9–2 も試みよ。
- 各課題について、プログラム・実行結果・それに関する説明を書くこと。
- さらに、課題を完了するのに要した時間と授業に対する感想も書くこと。
- 複数枚のレポートを提出する場合は、ホチキスでとじること。
- 他人の言説を**一部分**引用する場合は、どこが引用範囲であるか、また、出典がどこであるかが明確にせよ。断りなく他人の言説を書き写す行為は盗作になるので注意せよ。単にレポートの最後に「出典:○○○○」と記すだけでは、どこが引用範囲かを明確にしたとは言えない。また、他人の言説を長々と書き写す行為は、断りがあっても「引用」とは言えないことに注意せよ。

提出期限　2004 年 8 月 20 日 (金)

提出場所　情報教育棟レポートボックス

課題 9–1: (約数)　自分の学生証番号を n としたとき、1 以上 n 以下の n の約数を全て表示するプログラムを Java 言語で作り実行し、約数を調べよ。

例えば学生証番号が 987654 であれば、例えば以下のような出力をするものとする。(ただし、約数の個数が多い場合には、レポートには途中の出力を省略して、最初と最後の約数を 5 個程度ずつ含まれていればよい。また、表示の仕方はこの例に従う必要はない。)

```
My student ID is 987654, whose divisors are:
2
3
6
97
194
291
582
1697
3394
5091
10182
164609
329218
493827
987654
```

課題 9–2: (素数)　自分の学生証番号を n としたとき、n より大きい最小の素数を計算し、表示するプログラムを Java 言語で作り実行し、そのような素数を求めよ。例えば学生証番号が 987654 であれば、例えば以下のような出力をするものとする。

```
The smallest prime number above 987654 is 987659.
```

以上