目次

★1　オペレーティングシステムの概要

★1.1　オペレーティングシステムとは

現代のコンピュータシステムは，ハードウェアだけでなく**オペレーティングシステム** (**OS**) と呼ばれるソフトウェアがなければ動かない．OS は，コンピュータを使いやすくするために，かつ効率的に使えるようにするために存在しており，ユーザや**応用ソフトウェア**とハードウェアとの仲立ちをするものである．

OS の核となるのは，ハードウェアやユーザプログラムを管理する**制御プログラム**と呼ばれる部分である．狭義ではこの部分のみを OS と呼ぶが，広義では，プログラムの実行や作成に必要な道具（ソフトウェアライブラリやコンパイラ等）や，ファイル一覧ソフトやエディタ等の基本的なソフトウェアまで含めることもある (☆ 1)．

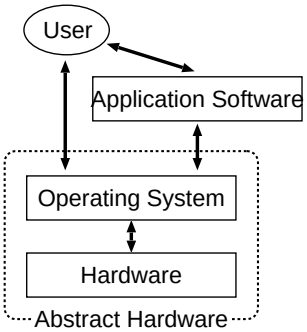
したがって，OS とそれ以外の部分との境界は曖昧である．例えば，PC の代表的な OS である Windows では，**GUI**(Graphical User Interface) の機能も OS と一体で提供されているので OS の機能と思いがちであるが，Linux 等では OS とは切り離されており，ユーザが好みの GUI をもつソフトェアを選択することができる．

応用ソフトウェアの実行プログラムは特定の OS 向けになっているので，通常は異なる OS のもとでは動作しない．

Operating System

応用ソフトウェア: Application Software, アプリケーションソフトウェアとも言う

☆ 1) 広義の OS を**基本ソフトウェア**と呼ぶこともある．また，基本—と応用—の中間的な位置づけのものを**ミドルウェア**と分類することもある（仮名漢字変換ソフト等）．



★1.2　オペレーティングシステムの役割

上述のように，OS の目的は主に使いやすさの向上とシステム性能の向上である．その役割には，次のようなものがある．

資源の管理: コンピュータシステムはプロセッサ，メモリ，外部記憶装置，プリンタ等のハードウェア資源と，プログラムやデータ等のソフトウェア資源から成っており，ユーザや応用ソフトはこれらの資源を要求する．OS は資源の状態を管理し，複数の要求が競合した場合でも適切な対応をする (☆ 2)

資源（リソース）: resource

☆ 2) 例えば，他のプログラムを待たせる，エラーを返す等．

コンピュータの利用効率の向上: 複数の処理を同時に行う際に，資源の割当順序等をうまく制御することで，全体としての処理のスループットを向上させる．

ハードウェアの抽象化:ユーザや応用ソフトが多種多様なハードウェアを直接操作しようとすると，個々のハードウェアの違いに対応したプログラムを作成しなければならない．OS はハードウェアの操作を抽象化した統一的なインタフェースを提供し，ユーザの操作や応用ソフトの開発作業を容易にする (☆ 3)．

☆ 3) 例えば UNIX では，様々な周辺機器から／への入出力をファイルの読み書きとして行えるようになっている．

★ 1.3 オペレーティングシステムの発展過程

OS の発展過程を概観しよう．

黎明期である 1940 年代のコンピュータには，OS は存在しなかった．ユーザがひとかたまりの処理（**ジョブ**という）をコンピュータに与えたり処理結果をコンピュータから取り出す際には人手を介していたため，一つのジョブを終えてから次のジョブを開始するまでの間，コンピュータは無駄に停止していた．

ジョブ: job

1950 年代になると，複数のジョブをまとめて連続的に処理する，**バッチ処理**の機能を備えた OS が開発された．これにより，コンピュータをジョブとジョブの間で停止させることなく効率的に利用できるようになった．

バッチ処理: batch processing

1960 年代には，より効率的にコンピュータを使用するための仕組みとして，**マルチプログラミング**という方式が開発された．また，同時に複数のユーザがコンピュータを対話的に使用できる**タイムシェアリングシステム**が開発された．1964 年には，これらの技術を取り入れた実用レベルの OS として，OS/360(☆ 4) が発表されている．この他，**仮想記憶**（次回以降説明します）の概念や，後の UNIX に大きな影響を与えた OS である Multics が登場したのもこの頃である．

マルチプログラミング: multi-programming，マルチタスク (multi-tasking) とも．

タイムシェアリングシステム: time-sharing system

☆ 4) OS/360 は，IBM の System/360 という汎用コンピュータに搭載された OS．コンピュータの発展に大きな影響を与えたものとして有名．

Multics はいくつもの革新的な技術を取り入れた大規模な OS であったが，開発に時間を要したこともあって普及するには至らなかった．この Multics のプロジェクトにかかわっていた技術者たちによって 1970 年前後に作られたのが **UNIX** である．設計がシンプルで柔軟である，異なるアーキテクチャのコンピュータへの移植が容易である (☆ 5) 等の理由により，広く使われるようになった．

☆ 5) それまでの OS はアセンブリ言語で書かれるのが普通で，UNIX も最初はそうだったが，すぐに C 言語で書き直された（C 言語はそのために作られた）．ちなみに，現在では UNIX は OS の名称ではなく仕様の名称となっている．

1970 年代後半以降，PC の登場にともなって様々な PC 向け OS が開発された．1980 年代には，MS-DOS, Mac OS, Microsoft Windows などが誕生している．

1980 年代後半以降，PC の性能向上にともない，UNIX のような OS（UNIX 系 OS）を PC 上で動作させようという取り組みがあちこちで行われるようになった．Linux(☆ 6) もそこから生まれた．

☆ 6) 1991 年当時フィンランドの大学生だった Linus Torvalds が一人で一から Linux（のカーネル）を作った．

★ 1.4 OS の機能の例

ここでは OS の機能の例として，複数のプログラムを並行して実行する方式，入出力の効率を上げる方式の 2 つを紹介する．

★ 1.4.1 複数のプログラムを並行して実行する

OS は，資源の有効利用のため，複数の処理を並行して実行できるようにする．その仕組みがマルチプログラミングとタイムシェアリングシステムである．

マルチプログラミングとは，複数のプログラムを同時にメモリ上に置き，CPU に実行させるプログラムを短い時間間隔で切り替える処理形態 (☆ 7) のことである．複数のプログラムを逐次実行する場合より資源を有効利用してスループットを向上させることをねらったものである．

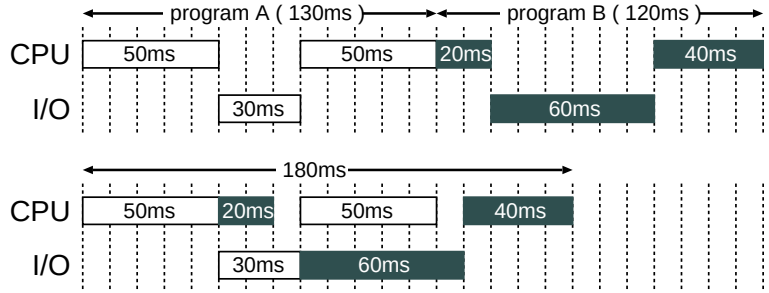
multiprogramming

☆ 7) このような処理形態を**並行処理** (concurrent processing) と言うこともある．並列処理との違いに注意．

例えば，CPU と入出力 (I/O) 装置を使用する 2 つのプログラム A,B の処理時間が下図上段のようになっていたとする．ここでは，CPU，I/O 装置とも一度に 1 つのプログラムだけが使用できるとしている．上段に示すように A の実行後に B を実行する場合，全体では 250ms の時間を要する．そのうち 90ms は CPU が何もしていない**アイドル時間**となっている．マルチプログラミングの形態でこ

アイドル時間: idle time.
idle a. 怠惰な，(機械などが) 休止状態である，v. 怠ける，空転する（エンジン等の空転状態を idling という）．idol とは別の語．

れらを実行する場合，例えば図下段のように実行することで，全体の処理時間を 180ms に，CPU のアイドル時間を 20ms に減らすことができる (☆ 8).



☆ 8) ここでは，B より A の方が優先順位が高いとして A の方を先に割り当てているが，逆に A より B の方が優先順位が高ければ B を先に割り当てることになる．このように，資源の割当においては，何らかの基準で優先順位を決める必要のあるケースが多い．

タイムシェアリングシステムとは，1 台のコンピュータを複数のユーザが共用しているのに，各ユーザがあたかもそのコンピュータを占有しているかのように使えるようにしたものである．各ユーザに一定の短い時間（タイムスライス）を割り当て，割当を次々切り替えていく (☆ 9) ため，ユーザからすると自分専用のコンピュータと対話しているような感覚でコンピュータを使用できる．このような対話型処理の実現により，コンピュータの利用形態は大きく変化した．

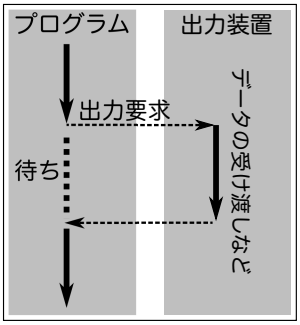
Time Sharing System(TSS)

☆ 9) このような処理を**時分割処理**ともいう

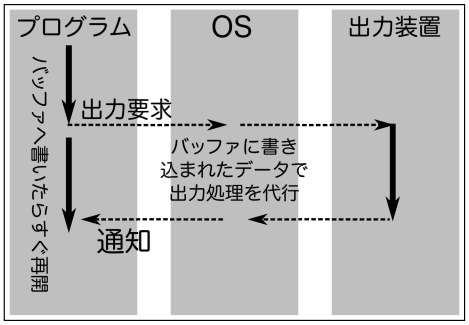
★ 1.4.2 入出力を効率化する

入出力を効率化する方式を 2 つ紹介する．ただし，ここで紹介するバッファリングやスプーリングは OS だけの機能ではなく，応用ソフトウェア自身が実現していることもある．

一般に，入出力装置を介した処理の速度は遅いため，プログラムの入出力をそのまま処理していると，図左に示すように待ち時間が発生して効率が悪い．これを解決する一つの方法は，入出力データを**バッファ**と呼ばれるメモリ内の領域にいったんためるようにして，入出力の処理と並行してプログラムを実行できるようにする，というものである．これを**バッファリング**という．



バッファリングなし



バッファリングあり

スプーリングとは，複数の周辺機器を同時に操作することを意味する．低速な周辺機器とのデータのやりとりを効率よく行うために，データをいったんメモリやディスク装置に書き込み，入出力の処理をプログラム本体の処理と切り離して実行できるようにする方式である．バッファリングではデータを主としてメモリ経由でやりとりするのに対して，スプーリングではディスク上のファイルを経由する場合が多いが，両者は類似した概念のものである．

スプーリングの代表例は，プリンタを接続しての印刷処理である．

バッファ: buffer，バッファリング: buffering，スプーリング: spooling (Simultaneous Peripheral Operations On-Line の頭文字をとった SPOOL を動詞化した語)

★ 1.5 カーネルとその機能

オペレーティングシステム (OS) の核となる部分のことを，**カーネル**という．狭義では OS とはカーネルのことを指す．前回説明した OS の役割（資源の管理，コンピュータの利用効率の向上，ハードウェアの抽象化）の大部分を担っている．

カーネルの機能には，主に次のようなものがある．

カーネル: **kernel**．**スーパーバイザ** (supervisor) と呼ぶこともある．前回資料の制御プログラムも同義．

［発展］ カーネルの構造／設計思想は，「モノリシックカーネル」と「マイクロカーネル」に大別できる．興味のある人は参考書やウェブ等で調べてみるとよい．Linux はモノリシック，Windows Vista や Mac OS X は（純粋なものではないが）マイクロカーネル．

プロセスの管理とスケジューリング ［詳しくは次回以降の授業で説明予定］

プロセス (☆ 10) とは，実行中のプログラムのことである．この範疇には，次のような仕事が含まれる．

(1) プロセス管理: プロセスを生成，消滅させる．プロセスの状態を把握する．
(2) スケジューリング: マルチプログラミングを採用したシステムでは，全体のスループットが高くなるように，複数のプロセスにプロセッサ（の処理時間）をうまく割り付ける必要がある．このような処理を**スケジューリング** (☆ 11) といい，カーネル内の**スケジューラ**と呼ばれる部分によって行われる．
(3) プロセスの同期と通信の制御: 複数のプロセスが並行して動作する場合，資源を共有するプロセスの間で，それらの処理が矛盾を起こさないようにタイミングを合わせる（同期をとる）必要がある．また，複数のプロセスが協調して動作できるように，**プロセス間通信**を実現しなければならない．

☆ 10) プロセス: process．**タスク** (task) と呼ぶこともある．UNIX 系 OS では，ps コマンドや top コマンドで実行中のプロセスを一覧することができる．

☆ 11) scheduling, scheduler. 実際にプロセッサの割り付けを行う作業のことをディスパッチ (dispatch) といい，それを行うプログラムをディスパッチャ(dispatcher) ということもある．

メモリの管理と仮想記憶の制御 ［詳しくは次回以降の授業で説明予定］

プログラムは，HDD 等の補助記憶装置からメモリ上にロードされてから実行される．メモリ管理とは，プログラムをいつロードするか，メモリ上にどのように割り付けるか，等を決める作業である．

マルチプログラミングシステムでは，複数のプロセスが同時に存在できるように，それらを同時にメモリ上に配置する必要がある．また，仮想記憶 (☆ 12) を採用したシステムでは，その制御も重要な仕事である．

☆ 12) 仮想記憶の方式を用いることで，プロセス全体をロードせず一部しかメモリ上に置いていないときでもその実行が可能となる．

入出力の制御と割り込みの制御 いずれもハードウェア（**デバイス**ともいう）の管理に関するものである．入出力制御とは，様々な入出力デバイスを管理して効率的に使用できるようにすることである．入出力制御のプログラムは，個々のデバイスに特化した部分（**デバイスドライバ** (☆ 13)）と，デバイスの種類によらない共通部分から成る．

割り込みについては次回以降の授業で説明する．

☆ 13) デバイスドライバ: device driver．主要なデバイスのためのドライバはあらかじめカーネルに組み込まれているが，周辺機器メーカが配布するドライバをユーザが自分で組み込むことも多い．

ファイルの管理 ［詳しくは次回以降の授業で説明予定］

OS の機能のうちファイルの管理に関するものは，**ファイルシステム**と呼ばれる．主として補助記憶装置上にファイルやディレクトリを構成し，それらの操作（作成，削除等）やファイルに格納されたデータへのアクセス手段を提供する．

ファイルシステム: file system.