第1章
オペレーティング・システムの働き

1.1　オペレーティング・システムとは

● オペレーティング・システムの誕生とその目的

オペレーティング・システム(Operating System；略してOS)が生まれたころ(当時は，まだOSとは呼ばれていなかったが)の「目的」は，コンピュータ・システム(ハードウェア)を最大限に効率よく運用することでした．しかし，現在は，ハードウェア価格の低下に加え個人利用も増加した結果，効率化よりもマン-マシン・インターフェース(または，ヒューマン・インターフェース)に重点を移し，コンピュータを道具として利用させようとするOSもあります(**図1.1**参照)．

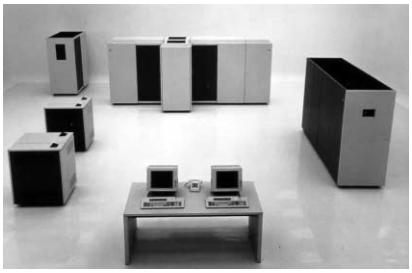
OSは，周囲の環境や**コンピュータ・システム**の目的に応じてその作動形態を変化させます．コンピュータの性能が低かった時代には，OSの汎用化はそれほど進んでいませんでした．その頃は，ある特定の環境に適し，ある特定の目的を達成させるためにOSが存在しました．

コンピュータの性能向上により，OSの汎用化が進み，あらゆる環境下で一つのOSが十分に能力を発揮できるようになりました．

また，コンピュータの稼働台数とその利用者の増加によって，「誰もが使える」という概念がOSに導入されました．OSとは，コンピュータという「物理的事象の集合体の物理現象」を論理的構成としてみせるからくりです．少し抽象的になったので，「料理」という作業を一つのアナロジとしてOSの意味を説明してみましょう．

●昔，コンピュータを使用する人は，ごく限られていた．
●このときのOSの目的は，できるだけ効率よくコンピュータを稼働させることであった．

(**a**) 昔のコンピュータ・システム(写真はIBM 360-20，1971年の機種)――当時は一般的に端末がつながらない．



●現在は，不特定多数の人がコンピュータを利用するようになった．
●このため，OSの目的は，マン-マシン・インターフェースの改善に焦点が移ってきている．

(**b**) 1990年前後のコンピュータ・システム(写真はIBM 3090E，1987年発表機種)

●各人が使うコンピュータとなっている．
●同等の物でサーバ機能も利用できる．

(**c**) 現在のパソコン

図1.1　コンピュータ・システムの過去と現在〔提供：日本IBM(株)〕

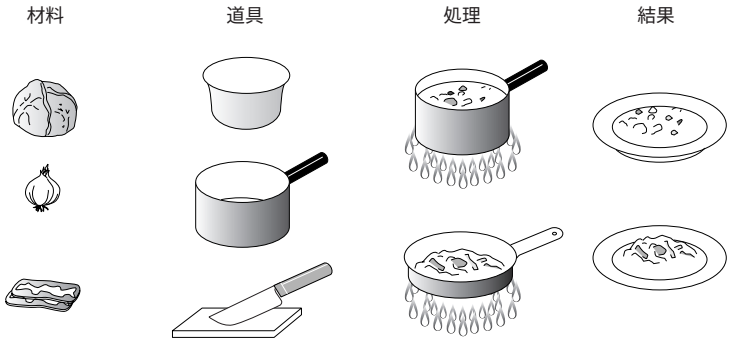


図1.2
料理を例にOSを考える

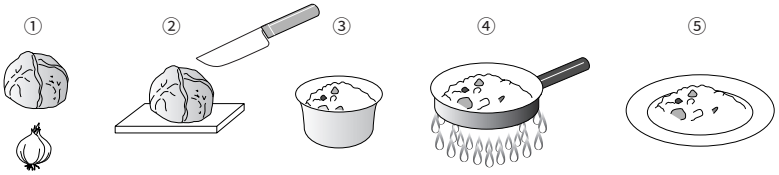


図1.3
逐次処理を行うOS

OSの仕事を，
(1) 原始的なOS……逐次的な処理
(2) 環境の把握　……利用できる道具の見きわめ
(3) 入力と出力　……材料と料理
の三点から見てみます．このアナロジにおいて，人間はコンピュータであり，器具や機械は入出力装置となります．そして材料が入力で，できあがった料理が出力です(**図1.2**)．

● 原始的なOS

上記(1)の原始的なOSの意味は，**図1.3**のように一連の料理の過程を想定して順番に進めることです．料理に必要な材料を，材料ごとに適した形に調え，加工し，最後に食べられるように盛り付けます．OSは材料ごとの加工方法や加工手段，味付けなどを指示する役割をします．

原始的なOSは，各作業を一つずつ逐次(シーケンシャル)に行います．より効果的に料理を行うためには，(2)の環境の把握が必要になります．

環境の把握とは，どんな道具があり，その使い方や能力を知ることです．OSはこの情報と料理の種類，材料などにより，料理の手順を組み立てます．

手順は原則的に**逐次処理**ですが，複数の加工が同時に進行する場合があります(**図1.4**)．

たとえば，ある材料を切ってから煮る場合，煮ている間に別の材料を切ることができます．CPUの場合では**図1.5**に示したように，CPUが処理1を起動した後はその処理1が終わるまで待つだけでよい場合，その間にCPUは処理2を起動できます．このように，複数の処理が(CPU以外のところで)行われるような場合を，**同時並行処理**(concurrent operation)と呼んでいます．

同時並行処理がうまく働くためには，OSがリソース(利用できる道具)を把握している必要があります．たとえば料理の場合，ボールや鍋が何個あって，レンジやオーブンが同時にいくつ使えるかなどです．これらを知っていることにより，料理の種類によっては複数の作業や複数の料理を同時に行うことができます．

OSが環境を把握していても，まだ効率の良い手順を決定できない要素があります．これは，(3)の入力と出力に関係します．料理のアナロジなら材料と料理の種類などです．つまり，どれだけの材料を使って，何をどれだ

●作業の効率を上げる方法として，自動的に進行する処理を同時に行う．
●火にかかっている間，別の作業ができる．

図1.4
同時実行処理とは

け作るかという問題です．

OSから見た問題とすれば，コンピュータが，何を，どのくらいの量を行うかということです．この問題は，OSに知らせる，または暗黙のうちに決めるなど，本来のOSの問題として扱われていません．

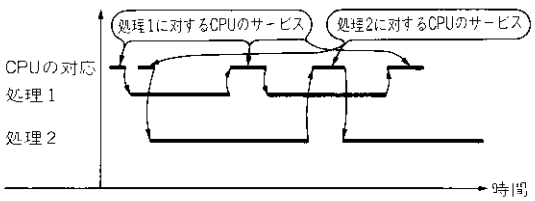
コンピュータが人間のみを相手にしているのであれば，これは“この計算にどの程度の時間が必要か”といった程度の問題です．しかし，コンピュータが制御を行っている場合などでは，10μs(0.00001秒)以内に処理しなさいなどと注文を付けられます．ハードウェアの性能が十分であれば，これが処理できるかどうかはOSの問題になります．

原始的なOSといわれるのは，これら(1)，(2)，(3)のすべてを，OSが稼働しているCPUが制御しているからです．

● 進んだOS

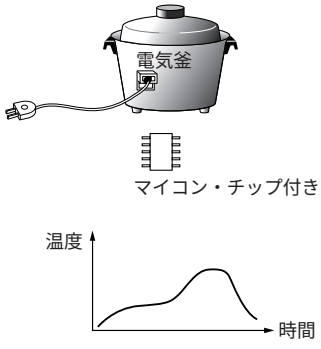
処理系が高機能化されるにともなって，OSが直接制御する仕事は減ります．料理の例では，米を炊く電気釜を考えてみてください．本来，火加減によってご飯の炊き上がりは変わってくるはずです．電気釜に与える電力を加減して炊き上がりを制御する電気釜があったとしましょう．OSが稼働しているCPUにより制御することも可能ですが(原始的なOS)，電気釜に組み込まれたマイコンが制御しても良いわけです．するとOSは，電気釜の制御から解放されます(**図1.6**)．この場合，OSは電気釜にご飯を炊けと命令するだけで済みます．この「ご飯を炊け」などという一連の仕事をタスク(task)と呼びます(プロセスと言う場合もある．**図1.7**)．

タスクとは，入力データ(米)と処理系＝プログラム(電気釜)が，ある結果(ご飯)を作り出すような仕事の単位



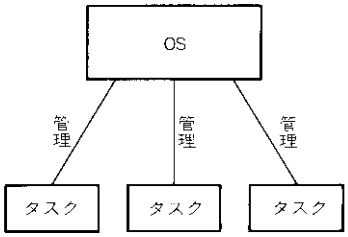
●**同時並行処理**とは，ある処理がCPUの介入なしに自動的に作動する時間を利用して，他の処理を並行的に行うことである．図では，処理1が起動された後に処理2が起動され，ある期間，処理1と2は並行的に進行する
●具体例を示すとRS-232-Cポートが9600ボーで8ビット・パリティ付き2ストップ・ビットのモードで動作しているとしよう．このとき1文字つまり8＋P＋2 stop＝11ビットのデータを受け取る時間は11÷9600　1.15msである．この間CPUは，何ら介入する必要はない．つまりこの間，CPUは別の仕事ができる

図1.5　同時並行処理の概念



●火加減はマイコンが制御する
●米を入れ，電気を入れるだけで理想に近い仕上がりになる
●OSはこの件には関与しない(電源ONのみ)

図1.6　高度なOSはレベルの低い作業が免除される



タスクとは，CPUから見た仕事の単位と考える．
図1.3の例では，②「加工①」，③「ためる」，④「加工②」，⑤「加工③」，があると考えられる．
この概念では，どこで仕事を区切るかで異なったタスクが作り得る．たとえば**図1.3**に対しては料理タスクという1個のタスクのみを想定することも可能である．
少し厳密にタスクを定義すると，「ある仕事のプログラムとそのデータの集合で，OSの処理の単位となり得るもの」となる．

- OSは，タスクを単位として全体の制御を行う
- タスクが何を行うかはユーザが決めることである．OSは，そのタスクの仕事内容には関知していない

図1.7　タスクの概念

と考えてください．

原始的なOSではタスク内の作業の制御までを行っていましたが，進んだOSではタスクに起動をかけ，終了を待つ形となります．したがって，この待ちの間に別のタスクの処理を行うことができます．タスクは，**図1.8**(**a**)のように完全に逐次的に実行される場合と，**図1.8**(**b**)のように複数のタスクが並行的に実行される場合があります．

複数のタスクが同時並行的に実行されるような制御を行うOSを，一般的にマルチタスクOSと呼んでいます．

タスクという概念を，一つの独立した仕事の単位と考えます．つまり，タスクは独立したデータとプログラムから成ります．互いに独立しているため，CPUがあるタスクの実行から別のタスクの実行に移る場合，すべてのCPUリソースを切り替える必要があるのです．

たとえば，タスクAからタスクBに切り替えるとき，タスクAは10番地の命令を実行しようとしていたとします．次にタスクAに実行が移ったときに10番地からプログラムを始められるよう，CPUの状態をタスクAに覚えさせておく必要があります．この，CPUのすべてのリソースを保全するというのは時間のかかる仕事です．

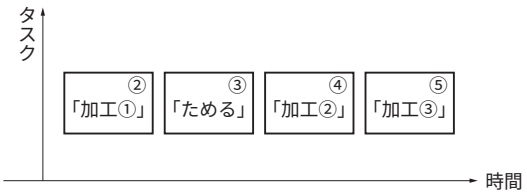
タスクの「互いに独立」であるという概念を「リソースのほとんどを共有する」というように言い換えれば，スレッドという概念になります．

鍋を使う料理で，魚(タスク1)と野菜(タスク2)を煮る作業において同じ1個の鍋を用いるとします．まず魚を煮た後，鍋を洗って野菜を煮ます．これがタスク切り替えです．リソース(鍋)を再利用できるような処理(洗う)が必要です．

スレッドは，処理系が何をするかわかっているときに利用できます．料理の例では，ステーキを焼いた(スレッド1)後のフライパンで野菜を炒める(スレッド2)などです(**図1.9**)．

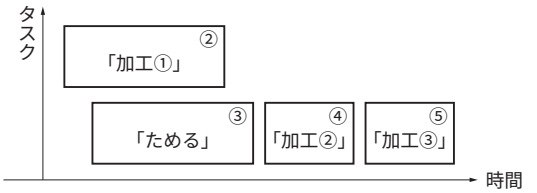
● ローカル・インテリジェンスとOS

図1.6のような電気釜を用いる場合，米をとぎ，電気釜にセットし，スイッチを入れるだけでご飯が炊けます．電気釜に組み込まれたマイコンがすべて制御します．このように，ある処理系が独自の判断機能(制御機能)をもっている場合，ローカル・インテリジェンス(local intelligence)があるといいます．



この場合，仕事は複数タスクで構成されるが，一時には1個のタスクのみが実行されている．この場合はマルチタスク処理とは言わない

(**a**) 逐次処理の場合



この例ではある時間，複数タスクを同時に実行している．このような処理を**マルチタスク処理**と言う．また，このような処理を可能にするOSを**マルチタスクOS**と言う

(**b**) 同時並列処理の場合

図1.8　マルチタスクと逐次処理

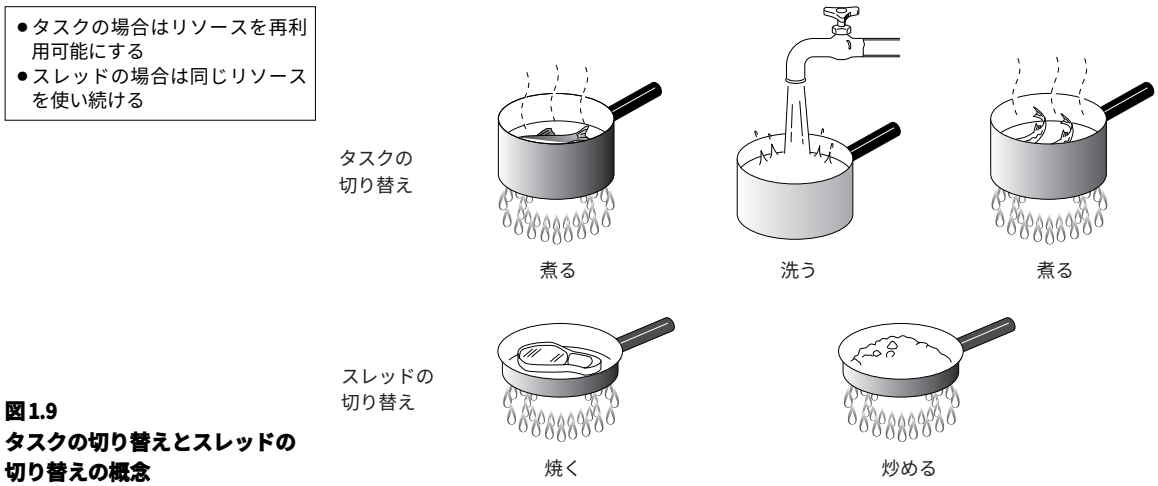


図1.9
タスクの切り替えとスレッドの切り替えの概念

この種の装置を管理するOSの仕事は，

① 状態を調べる

② 指示を出す

ことになります．

あるコンピュータ・システムにおいて，あらゆるタスクがローカルに，しかもインテリジェンスをもった処理系で実行されるような環境はありません．しかしコンピュータ・システム群を統括するようなシステムでは，ほとんどがこの手法になります．たとえば，パーソナル・コンピュータを連結してある仕事をさせる場合などがこれに相当します．

リソース内に複数のCPUが含まれている場合があります(マルチCPUシステム)．この場合，OSはさらに複雑になります．複数のCPUがOSの仕事を行うケースも配慮する必要があるからです．

1.2　OSの役割

● OSの意味

OSは，広い意味でコンピュータ・システムの運転全域に関与します．つまり，OSは，

(1) 人間とコンピュータ・システムとのインターフェース

(2) アプリケーション・ソフトウェアに対する資源の解放

(3) コンピュータ・システム間のインターフェース

(4) コンピュータ・システムに接続されている資源の有効利用

(5) コンピュータ・システムの変更や新しい機能の追加手段

などの機能を含みます．

狭い意味でのOSは，おもにコンピュータ・システムの資源の有効利用の問題に帰着します．

実際のところ，OSの定義は，はなはだ曖昧なものです．本書では，「コンピュータの基本的な性質を定めるプログラム群」としておきましょう．

コンピュータは，人間ほど複雑怪奇にはできていませんが，OSは人間の行動の規範を司る部分に似ていると考えられます．つまり，同じコンピュータのハードウェアであっても，OSが異なると同じ刺激に対して異なった反応を示します．

● OSの二つの側面

OSは，コンピュータ・システム内で二面性をもっています(**図1.10**)．

その一面は，ユーザ側を向いた顔といえるもので，可能な限りユーザにとって融通のきく構造をとります．こ