1.2 オペレーティングシステム（OS）

ここでは、サーバーで利用されるオペレーティングシステム（Operating System ：OS）の機能や役割について説明します。サーバーはハードウェアとOSおよびアプリケーションソフトウェア（アプリケーション）により構成されています。多くのクライアントからの要求を処理するために、ハードウェアとOSの関わりは非常に重要になってきます。ハードウェアを選定する場合、どのOSを利用するかも考慮する必要があります。サーバーで利用されるOSは、Windows ServerやLinuxなど用途に応じて使い分けられますが、ここではOSの一般的な機能を説明します。

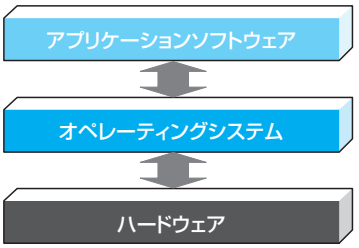

図1.3　オペレーティングシステムの位置づけ

1.2.1　OSの機能

OSの機能は主に3つあり、1つ目がハードウェアの抽象化です。例えば、何かを印刷する場合、異なるメーカーのプリンターでも同じように印刷することができます。これは、OSがプリンターの仕様の違いを吸収し、どのようなメーカーでも同様に扱える仕組みをアプリケーションに提供しているからです。

2つ目はリソース（資源）の管理です。複数のアプリケーションが動作する時に、それぞれのアプリケーションが利用するメモリの割り当てや、ハードディスクへの同時アクセスがあっても矛盾なくデータの取り出しができように制御します。

3つ目はコンピューターの利用効率を向上させることです。サーバーには多くのクライアントが一度にアクセスする可能性があります。処理を最大限速く実行して、クライアントに応答を返す必要があります。したがって、OSは処理を最適化するために、複数のアプリケーションの優先付けを行い、さらに複数のCPUを搭載している場合は、どのCPUにどのような作業を割り振るかなどの管理を行ないます。

コラム：JIS規格におけるOS

WindowsやLinux、Mac OSは代表的なOSですが、パソコンに搭載されるOSだけがOSではありません。パソコンのOSは汎用性を重視するために、複雑でサイズが大きく、そして多機能なのが特徴で、一般的なパソコンでは普段使われない機能も実装されています。それに対し、電化製品や電子機器にはコンパクトなOSが組み込まれています。それらは、サイズが小さく機能も限定されています。例えば、電子レンジには温度や時間を調整する機能だけを持つOSが組み込まれています。パソコンのOSと比べれば汎用性はありませんがこれらも立派なOSです。

種々の処理形態の要求を満たすことがOSの目的ですので、OSによって提供する機能も大きく異なります。したがって、実はOSの定義は明確ではありません。

ただし、JIS規格JISX0001上は、「プログラムの実行を制御するソフトウェアであって、資源割振り、スケジューリング、入出力制御、データ管理などのサービスを提供するもの」と定義されています。

1.2.2 カーネルの機能

OSは一般的には複数のアプリケーションやユーティリティソフトウェアを含めてOSと呼んでいますが、カーネル（Kernel）はOSの中心的な処理を行なう部分です。図1.4にカーネルの構造を示します。

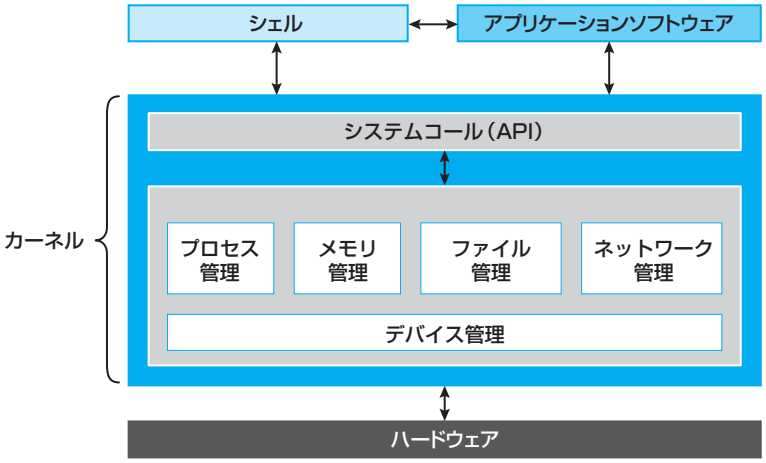

図1.4　カーネルの構造

NOTE

ユーティリティソフトウェア：OSやアプリケーションなどを補助するためのソフトウェアです。

OSが持つ基本的な機能はカーネルが受け持っています。カーネルは主に以下の処理を行ないます。

- ハードウェアの抽象化
- リソースの管理（メモリ管理・デバイス管理・ファイル管理・ネットワーク管理）
- コンピューターの利用効率向上（プロセス管理）

コンピューター内のリソースを管理し、メモリ、CPU、入出力装置を中心としたハードウェアを抽象化し、アプリケーションに共通のインターフェイスを提供します。また、アプリケーションへ提供する機能として、プロセスの抽象化、プロセス間通信、システムコールなどの処理があります。プロセスとは処理の実行単位で、アプリケーションを実行する場合に、カーネルがプロセスを作ります。カーネルは、プロセスにメモリやCPUの実行時間を割り当てたり、複数のプロセスを並行して処理することを可能にしたりします。この処理はカーネルによって方式が異なり、設計や実装なども異なっています。多くの場合、カーネル自身はプロセスとしては存在せず、ディスクアクセスなど優先度の高い処理が発生したときなどにアプリケーションから呼び出され、カーネルの処理が終了するとアプリケーションに戻ってくる仕組みになっています。

■ プロセス管理

カーネルはアプリケーションを実行するため、ハードディスクにあるプログラムを読み出してメモリに配置します。その後、プログラムに処理を渡すわけですが、その実行単位をプロセスと呼びます。カーネルはプログラムにメモリやCPUを一定時間割り当てます。複数のプロセスが存在する場合、カーネルは実行するプロセスを一定時間ごとに切替えて、あたかも同時に処理をしているように見せています。これをマルチタスクといいます。プロセスを切替える仕組みは、スケジューラーという制御プログラムが行ないます。プロセスの優先度を考えながら各プロセスに時間配分を行ない、優先度の高い処理がより多くCPUを占有できるようにします。スケジューラーの時間配分アルゴリズムはOSによって異なる場合があります。

各プロセスは、プロセスの独立性を高めるためにメモリ上のデータを共有することができません。その為、プロセス間で情報のやり取りを行なうプロセス間通信（IPC）と呼ばれる仕組みを提供しています。プロセスをさらに分割した実行単位にスレッドがあります。プロセスはスレッドを複数束ねたものですが、スレッド間ではメモリを共有して利用する事ができます。プロセス内で複数の処理を同時に実行させ、さらにメモリ共有したい場合にスレッドが用いられます。例えば、ワープロで文書のチェック処理を実行させながら文字の入力を行なう場合、チェック処理と入力処理を別々のスレッドにし、文章を共通のメモリ上に置くこ

とで、同時処理が可能となります。

サーバーがクライアントからの要求を受けた時、1つの要求に対してプロセスまたはスレッドを1つ割り当てて処理を行なう場合があります。プロセスまたはスレッドを割り当てないと複数のクライアントの処理を並行して実行できないからです。CGIの処理がこれにあたります。アプリケーションによってプロセスやスレッドを利用するケースは変わってきますが、1つのプロセスまたはスレッドによってメモリなどのリソースが使われることになるため、プロセスやスレッドが最大でいくつ作られるのかを事前に把握しておく必要があります。

また、複数のCPUに対応したOSがありますが、スレッド単位で処理を分散しており、単一のCPUより高速処理を実現できます。LinuxやWindowsなどは複数のCPUに対応しているので、アクセスが集中するサーバーで利用するのは有用です。

プロセス管理においては図1.5のように、各プロセスが状態を遷移します。カーネルは優先順位をスケジュール管理しながらプロセスにCPUを割り当てます（プロセス実行状態）。より優先順位の高いプロセスが生成された場合は、実行中のプロセスを実行可能状態に遷移させ、優先順位の高いプロセスを先に実行させるなどの処理を行ないます。実行中のプロセスは入出力待ちなどにより待ち状態になることがあります。入出力が終了すると、実行可能状態になりCPUが割り当てられるのを待ちます。

プログラムの起動でプロセスの生成を行い、プログラムが終了すればプロセスを終了させます。

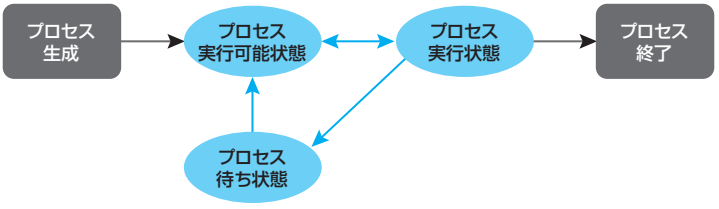

図1.5　プロセス管理

■ メモリ管理

カーネルはプロセス生成時またはプロセスからの要求に対してメモリを割り当て、不要になったら利用しているメモリを解放します。解放とは、他のプロセスで利用できるようにすることです。プロセスはメモリを勝手に利用することはできませんので、カーネルにメモリ

NOTE

CGI：プログラムとWebサーバーを連携させる仕組みで、動的Webページを生成します。

を利用する旨を依頼する必要があります。この仕組みにより、各プロセスで利用するメモリを間違って他のプロセスが書き換えてしまうことがないようにしているわけです。もしあるプロセスから違うプロセスのメモリの変更が可能であれば、1 つのアプリケーションのバグによりコンピューター全体が止まってしまう可能性があります。

メモリ管理のもう1つ重要な機能は、仮想記憶の実現です。その方式は次の2つがあります。

- スワッピング方式
- ページング方式

スワッピング方式は、ハードディスク上のプログラムやデータを読み出すタイミングでメモリが不足していた場合、その時点で使用されていないプログラムやデータをメモリからハードディスクへ移動して空領域を作った後、必要なプログラムやデータを読み出します。物理メモリとハードディスク間で入れ替えを行ないますので、スワッピング（交換）と呼んでいます。

ページング方式は、物理メモリを一定のサイズで分割します。分割した単位をページと呼んでいますが、このページ単位でハードディスクに退避および読み出しを行ないます。物理メモリに必要なプログラムやデータを読み出す際にメモリが不足した場合、その時点で使用されていないページをハードディスクに退避して空領域を作った後、必要なデータをメモリに読み出します。

スワッピング方式もページング方式も不足する物理メモリを大きく見せるための仕組みではありますが、メモリとハードディスクのアクセス速度（読み出し時間または書き込み時間）に大きな違いがあります。少ない物理メモリで大きなデータを扱う処理を行なった場合、ハードディスクへのアクセスが頻繁に発生してしまい処理が大幅に遅くなってしまいます。このような性能低下のことをスラッシングといいます。多くのクライアントから要求があるサーバーを、少ないメモリで運用しているとスラッシングが多発してパフォーマンスの悪いシステムになってしまいますので、ピーク時にいくつのクライアントからアクセスがあるかなどを事前に予測して、必要なメモリなどを計算しサーバーに実装しておく必要があります。

図1.6にスワッピング方式とページング方式の違いを示します。

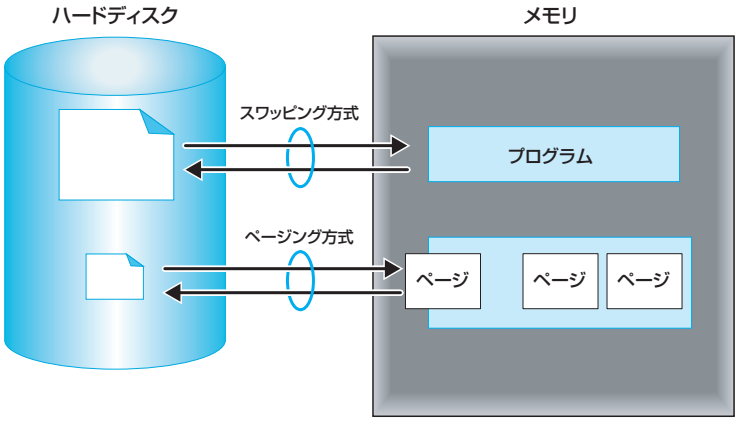

図1.6　スワッピング方式とページング方式

■ デバイス管理

ディスプレイやプリンターへの出力などコンピューターは周辺装置とのデータのやり取りが必要になります。周辺装置とのデータのやり取りには周辺装置のメーカーが開発した、デバイスドライバーと呼ばれるプログラムが必要です。デバイスドライバーはカーネルの一部として機能しますので、アプリケーションからハードウェアを直接制御する必要がなくなります。デバイスドライバーがハードウェアを抽象化しているので、異なるハードウェアでも同じアプリケーションを利用することができ、また同じハードウェアでも違うOSで利用することができるわけです。

デバイスドライバーはカーネルに組み込まれて動作するために非常に重要なプログラムの一つといえますが、まれに不具合が発生します。サーバーなどで取り扱う周辺装置は導入や使用実績のあるものを利用するか、事前に十分検証する必要があります。

■ ファイル管理

ファイル管理とは、ハードディスクなどの補助記憶装置に対するデータの読み書きや、フォルダーによるファイルの保管を行うことです。ファイル管理方法はOSにより異なり、ファイル管理全体をファイルシステムと呼んでいます。実際は物理的な管理と論理的な管理に分かれます。物理的な管理とは補助記憶装置のどこにデータを配置するかなどを管理することです。ハードディスクであればトラックやセクターという単位で区切られて保存されますが、1つのファイルが連続したトラックやセクターに配置されるとは限りません。ばらばらに配置されていても、連続して読み込む必要がありますので、ハードディスク上のどこに保

存されているかを物理的に管理する必要があります。一方、論理的な管理とは、フォルダーを利用した階層構造などを管理することです。階層構造として管理していないと、多量のファイルから目的のファイルを探すのに時間を要してしまいます。階層構造にすることにより、ユーザーにとって分かりやすい管理が行なえます。

ファイルシステムは利用できるファイル名の命名規則や扱う文字コードなどをそれぞれ決めているので、ファイルシステムが違うとファイルの共有が出来ない場合があります。そのようなケースでは専用のソフトウェアが必要になります。

コラム：OSとデフラグ

デフラグとは？　ファイルの断片化によってばらばらに配置されたものを整理し直す機能です。断片化（フラグメンテーション）を取り除く「デ・フラグメント」の略でデフラグといいます。

それではファイルの断片化とは、どのようなものでしょうか？

ハードディスクの中身を見てみると下図のようにデータを書き込む部分は、決められた容量を持つ入れ物（区画）の集合となっています。

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20

この入れ物にファイルA（4区画必要）、ファイルB（3区画必要）、ファイルC（5区画必要）を書き込みます。（下図）

A	A	A	A	B	B	B	C	C	C
C	C	13	14	15	16	17	18	19	20

次に、ファイルBが不用になったので削除しました。さらに、ファイルD（6区画必要）を書き込むとどうなるでしょうか？（下図）

A	A	A	A	D	D	D	C	C	C
C	C	D	D	D	16	17	18	19	20

このようにファイルDは2つの区画に分かれてしまいます。これが断片化です。断片化が起こると、ハードディスクのデータを読み書きするヘッドが連続で動かせないので、読み書きに時間がかかってしまうことになります。

Windowsで利用されているファイルシステムでは断片化が起きてしまいますが、Linuxで利用しているファイルシステムでは起こりにくいといわれています。

どうしたら、断片化を起こりにくくすることができると思いますか？
答えは単純です。ファイルの間隔を充分あけて書き込めば良いのです。

A	A	A	A	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
B	B	B	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
C	C	C	C	C	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60

この状態からファイルBを削除して、ファイルDを追加しても下図の通り断片化は発生しません。

A	A	A	A	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
D	D	D	D	D	D	27	28	29	30
31	32	33	34	35	36	37	38	39	40
C	C	C	C	C	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60

このように断片化の発生頻度はOSによってさまざまです。

■ ネットワーク管理

ネットワークデバイスの制御や通信プロトコルの機能を提供します。ネットワークに接続するためのNIC（Network Interface Card）や無線機器の制御を行なったり、インターネットで利用されているTCP/IPプロトコルの提供を行なったり、またその管理を行ないます。ネットワーク経由でディスク資源の提供も行ないます。UNIX／LinuxではNFS（Network File System）、WindowsではSMB（Server Message Block）という仕組みによりファイルやフォルダーの共有が行なえます。

■ システムコール

アプリケーションからカーネルが提供するサービスを要求する時にシステムコールを利用します。例えばメモリを確保する時やファイルへアクセスする時などが挙げられます。アプリケーションプログラムはカーネルに対して勝手にアクセスすることはできません。カーネルが破壊されると、コンピューターシステムとして正常に動作しなくなるため、システムコールという決められた手順によってカーネルにアクセスします。

1.2.3 シェルの機能

シェルとはカーネルとユーザーとの会話を行なうためのソフトウェアで、カーネルへのユーザーインターフェイスを提供します。カーネル自身はユーザーと直接会話をする機能を持っていないために、このシェルが必要になります。シェルはコマンドを解析してカーネルに渡します。コマンドに間違いがあった場合は、シェルがエラーメッセージをユーザーに返すことになります。ユーザーから見てカーネルを包み込む様子が、あたかも「殻（shell）」のように見えるのでシェルと呼ばれるようになりました。

UNIX / LinuxなどはCLI（コマンド・ライン・インターフェイス）が一般的で、キーボードからのコマンド入力を行ないシェルに渡します。Windowsの場合は、cmdがCLIのシェルに相当しますが、WindowsはGUI（グラフィカル・ユーザー・インターフェイス）がシェルの役目も担っており、マウスの操作だけでもシェルに指令を送ることができます。

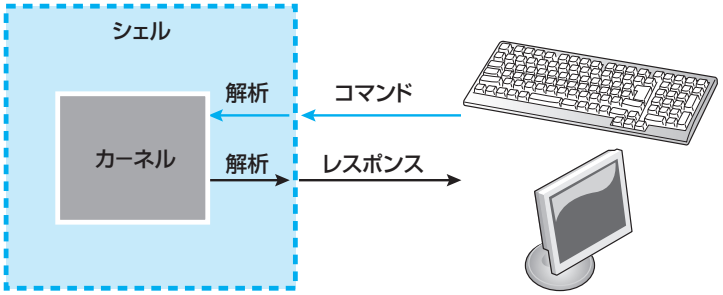

図1.7　カーネルとシェルの関係

コラム：モノリシックカーネルとマイクロカーネル

カーネルは構造の違いからモノリシックカーネルとマイクロカーネルに分かれます。モノリシックカーネルとはカーネルにネットワーク機能や入出力機能など、さまざまな機能を実装してまとめたものです。それに対してマイクロカーネルはカーネルに機能を絞って実装して、それ以外の機能は別のモジュールとして動作させます。モノリシックカーネルは全ての機能が一体化しているのでメモリの効率が悪くなりますが、高い処理速度が得られます。ただし、新たに機能を追加しようとした時に全体を再構築する必要があります。マイクロカーネルの機能の追加は、モジュールの追加で済むメリットがあります。

モノリシックカーネルの代表的なOSはLinuxで、マイクロカーネルの代表的なOSにはWindowsがあります。