Raise/Linux カーネルの教科書

Linuxカーネルとは

- OSの役割
  - キー入力やマウスクリック等の操作インターフェース
  - アプリケーションが使う共通機能
  - ハードウェア制御
- カーネルとは
  - OSの中核的な役割を果たす
  - OSを植物の種子に見立て、カーネルを種子の内部「仁(kernel)」、インターフェースを種子の「殻(shell)」としている。
  - OSによって担当する機能は異なる
  - Linuxカーネルはファイルシステムやデバイスドライバも含んでいて、比較的範囲が広い
- Linux
  - アプリケーションがカーネル配下の機能を使ったり、ハードウェア資源を操作する際に使用するシステムコールがUNIXとほぼ共通
  - UNIXを移植性の高いプログラミング言語でコンパクトに実装した。
    - この流れでUNIX派生が多くでき、その中でOS間のアプリケーションの互換性を保つためにPOSIXで共通インターフェース規格が定められた。
  - Linuxは本来カーネル部分のみのことを指す
  - OSとして扱われているのはLinuxディストリビューション
    - カーネル以外をGNUプロジェクトというフリーなUNIX互換システムを利用していることからGNU/Linuxシステムともいう

イベント駆動型

- イベント駆動型
  - Linuxカーネルは実行コードは常時メモリにある
  - 通常は実行されない
  - ハードウェアやアプリから何らかの指令があったら実行される
  - イベントが動作の引き金になるようなプログラムをイベント駆動型プログラムという
  - カーネル処理においては当該のイベントが3種類ある
    - 周辺機器からCPUへの「**ハードウェア割込み**」
    - ソフトの動作によるCPU内での「**ソフトウェア割込み/例外**」
    - アプリが発行する「**システムコール**」
      - アプリが実施できないハードウェアの直接制御やカーネルの動作制御

割り込み

現在処理中の作業を一時中断して別の処理を実施すること

割り込みが発生するとCPUは

記憶装置の特定箇所にある「割り込みベクター」を参照

↑で競ってされているアドレスにあるプログラム「割り込みハンドラー」を実行

↑の末尾に割り込み処理終了の命令があり、それによって元の処理を再開

システムコールの呼び出しにも使われていた

発行はアプリが行うが、処理はカーネルが行うため、この実行主体の切り替えにソフトウェア割込みが用いられていた

処理が遅い

今は割り込みなしで出来るようにしている

タスク管理

プロセスとスレッド

Linuxでプログラムを実行するとプロセスまたはスレッドが作成される

プロセス

主メモリーに読み込まれ、独立したメモリー空間を割り当てられて稼働中のプログラム

スレッド

他のプロセスやスレッドとメモリー空間を共有する特殊なプロセス

カーネルの動作を補助するカーネルスレッドなどもある

プロセスやそれっどには個別のプロセスIDが割り振られている

タスクスケジューラー

実行待ち状態のタスク群から、どのタスクをどのくらいの期間、どのCPUで実行するのかを管理する

各タスクを公平にかつ効率よく実行することが求められる

タスク

プロセスやスレッド

カーネルがCPUで実行するタスクを短い間隔で切り替えることで複数のタスクを同時に展開できる

これを並行処理と呼ぶ

Linuxでは各タスクがCPUを使用した時間を計測して、それが短いタスクから実行するCFSを採用

メモリ管理

仮想アドレス空間

Linuxはマルチユーザー利用にも対応しているため、各プロセスの利用するメモリー領域が被ったり、各ユーザーのプロセスのデータが見れるような状態にならないようにする必要がある

各プロセスに独立したアドレスを割り振ったアドレス空間で動作させる
主メモリではない
これを仮想アドレス空間という
実際に利用する場合は物理アドレス空間にマッピングする必要がある

ページング方式
仮想アドレス空間と物理アドレス空間のマッピング方法の一つ
メモリー領域をページの集合として管理する方式
ページテーブルを設定することで自動的なアドレス変換を行う
一般的にはCPU内の制御レジスタにページテーブルの物理アドレスなどをセットしておく
プロセス生成時にそのプロセス用のページテーブルを作成する
多くのCPUは複数段のテーブルを作成することでテーブルによるメモリの使用量を削減している
入出力などが発生した際にのみ物理アドレス空間へのマッピングを行う
デマンドページング方式という
物理ページ未設定の下層ページにアクセスしたときの例外を用いて実装

メモリースワッピング
空き物理メモリが少なくなると最近使用していないページを回収する
ページキャッシュに利用している物理ページの内容を破棄して秋物理ページにする
変更されているファイルを定期的にファイルに書き戻している
プロセスが作業領域などのために利用している無名ページをハードディスクやファイル上のスワップ領域に書き出す
ページアウト
物理ページがページアウトされている下層ページにアクセスがあった場合は、中らな物理ページが割り当てられ、そこにページアウトされたデータが書き戻されて使用される
ページイン
ページインとページアウトをまとめてメモリースワッピングと言う
システム全体でブリメモリーの容量以上のメモリを使用できる
時間がかかる
メモリの使用量がページ回収によって解決できない場合は、OOM Killerによって使用中の物理ページや、ページアウトしているページの多いプロセスを強制終了することで、メモリを確保する

デバイス管理

デバイスファイル
Linuxではほぼすべてのデバイスをファイルとして抽象化する
アプリからデバイスにデータを入出力する場合は、このファイルに対して入出力をする
デバイスファイル

一般的には/devにある

デバイスを2種類に分けている

キャラクター型

シリアル回線や端末の様にデータを1バイトずつ入出力する

データは基本的にバッファーされない

ブロック型

ハードディスクの様に固定長のデータ単位に入出力する

データは基本的にバッファーされる

デバイスファイルには型を示す情報が設定される

デバイスに応じた「メジャー番号」「マイナー番号」が設定される

メジャー番号

デバイス制御用のデバイスドライバの指定に用いる

マイナー番号

デバイスドライバの制御化にある個々の機器を指定するのに用いる

カーネルはこれらの情報に基づいて、デバイスドライバとデバイスファイルを紐づけする

ファイルシステム

ファイルシステム

「ファイル」というインターフェースをユーザーアプリケーションに提供する仕組み

記憶装置にデータがどのように記録されているかを意識することなく、記憶装置の種類の違いを意識することなくデータの入出力が出来るようになる

「ファイルの管理性の向上」「記録データのセキュリティの確保」のための機能も提供している

ディレクトリ・権限など

記録データの信頼性を確保するために、データ検証用のチェックサムを記録できるものもある

ファイルシステムが異なっていても、ファイルシステムの違いを吸収して、統一したインターフェースを提供する抽象化層VFS(virtual file system)が用意されているため、ファイルの読み書きの方法は共通である

ネットワーク機能

通信プロトコル

ソケットと言うインターフェースを利用することで、様々ン通信プロトコルに対応した柔軟な通信が可能となる

ソケットインターフェースを提供するための抽象化層(BSDソケット層)が存在する

LinuxカーネルにおいてはIPやその上位のTCP/UDP、IPXや内部通信用の特殊なプロトコルなどをサポートしている

ソケットを利用する際に、どのようなプロトコルグループを使用するのか指定する

Linuxカーネルには Netfilter という通信データ処理用の気候が存在し、これによって特定のパケットの送受信を制限したり、加工した処理が可能となる

Linuxのファイアウォール機能やNAPT機能に使われてい

NAPT

ポート番号情報を併用することで1対多のアドレス変換に対応したNAT技術の一つ

読み込みから稼働まで

稼働順序

PCの電源投入直後にBIOS(Basic I/O System)や、UEFI(Unified Extensible Firmware Interface)が稼働し、PCのハードウェアの初期設定や自己診断テストを実施した後、ディスクの特定領域に記録されているOS起動用プログラムのブートローダーを起動する

LinuxではGNU GRUBというブートローダーが使われている

ブートローダーがカーネルのイメージファイルをメモリに読み込んで実行する

カーネルのイメージファイルはbzImageという自己伸長圧縮形式になっている

イメージの先頭には、初期化とイメージ伸長のためのコードが付加されている

初期化ディスクイメージ

多くのディストリビューションでは、起動時にシステム初期化用のディスクイメージを読み込む

デスクイメージはシステム初期化の際に、rootファイルシステムの読み出しに必要デバイスドライバがカーネルイメージに組み込まれておらず、モジュールファイルとして切り離されている場合に必要となる