第 5 回オペレーティングシステム

スレッド

プロセス内での並行処理を実現する仕組み→スレッド　（経路プロセス）

プロセス＝異なる目的のタスク　スレッド　＝同じ目的のタスク

プロセス内の資源（データなど）を共有しながら処理の見えお多重化捨
コンテキストは切り替えるが、資源情報は切り替えない
入れ替える情報量が少ないので低コストで切り替えできる

ユーザレベルスレッド（OS 介入なし）とカーネルレベルスレッド（OS の介入あり）がある。OS が介入すると切り替えの安定感は増すが、切り替えのオーバーヘッドも増える。

OS によって用意されている仕組みが異なるので注意

まとめ

プロセスは仮想的なプロセッサとプログラムのリンク

プログラム管理の方法：プロセス記述し
プロセスの状態は 3 つ：
実行しているけど処理してない状態

スケジューリングによるプロセッサの効率的利用

プロセス間通信

プロセス間の協調動作

プロセスはプロセッサも全て独立
→他のプロセスがどのような状況でも動作を保証

作業の効率化には情報共有した上での並列作業
→他のプロセスの作業情報なしでは同じ作業を繰り返す

他のプロセスと状況を教えあう仕組み
→プロセス間通信
　IPC (Inter Process communication)

作業の手分け

P35 参照

プロセス間の連絡手段

シグナル
共有メモリ
パイプ
ソケット

シグナル

プロセスにイベント通知を行う方法

シグナルを受け取ったプロセスは処理をやめ
シグナルで決められた処理（シグナルハンドラ）を実行する
教科書 P35 図 4.2 参照

コマンドライン Tips

プロセスにシグナルを送る

Kill コマンド

Kill -9 1234
→1234 のプロセスを強制終了

Kill-HUP 1234
→1234 番のプロセスを再起動

Kill-1
→シグナルのリストを表示

共有メモリ

プロセス間でデータ共有をするための基本的な方法

互いのプロセスが持つメモリを空間の一部を同じ物理メモリで共用

１方で書き込んだ内容は、もう１方にも反映さえる。

パイプ

プロセスの出力と入力を繋ぐための仕組み

一方のプロセスの出力結果をもう１方の入力データとして利用できる

プロセスの間に FIFO のバッファ作られる
教科書 P37 図(a)参照

コマンドライン Tips

コマンド間をパイプで連結

パイプ　“|”（縦棒）

例：　ls | tee lstest.txi
　　→ファイルリスト表示(ls)した内容を lstest.txi にファイル保存
(tee)
例：[s anx | grep user1

→プロセスリスト表示(ps)した内容から user1 というキーワード
が入っている行を抽出して表示(grep)

ソケット

同一マシン上で動作していないプロセス間での通信に使用（同一マシン内でも可）

ネットワークを介して、それぞれのホストが持つバッファの読み書きを行う

信頼性の高い TCP ソケットや遅延の少ない UDP ソケットがある
P37 図(b)参照

競合状態と相互排他

複数のプロセスが共有するデータには注意が必要

同じデータの書き込みが同じタイミングで発生

書き込んでいる途中で読み込みが進行

読み込んだデータを処理して書き込む途中で他者がデータ更新

→競合状態

競合の危険性

P38 図 4.5 参照

相互排他

競合状態をなくすには

とラウルになりそうな場合は
対象の作業が終わるまで他のプロセスに干渉させない

資源を１つのプロセスに占有させるための制御
→相互排他または排他制御

競合状態が起きないような仕組み

際どい領域

クリティカルリージョン　クリティカルセクション

他のプロセスが干渉すると問題になりそうな処理のこと

この領域の処理に競合が発生しなければ、システムは健全に動作
いくつかの条件を保つことが必要

際どい領域の処理が常にプロセスに限定されれば問題ない
→問題なし

際どい領域の処理を複数のプロセスが実行しようとした場合
→１つを残して制止（ブロック）

協調動作のための条件

健全な制御が行われるためには

必要ないとこ路で他のプロセスをブロックしない→条件２

プロセスによってえこひいきしない→条件３

いつまで待っても順番が来ないのはだめ→条件４

P39 図 4.6 参照