システムプログラム概論

Introduction to Systems Programming

スレッドとコンカレンシ

門林　雄基

Youki Kadobayashi

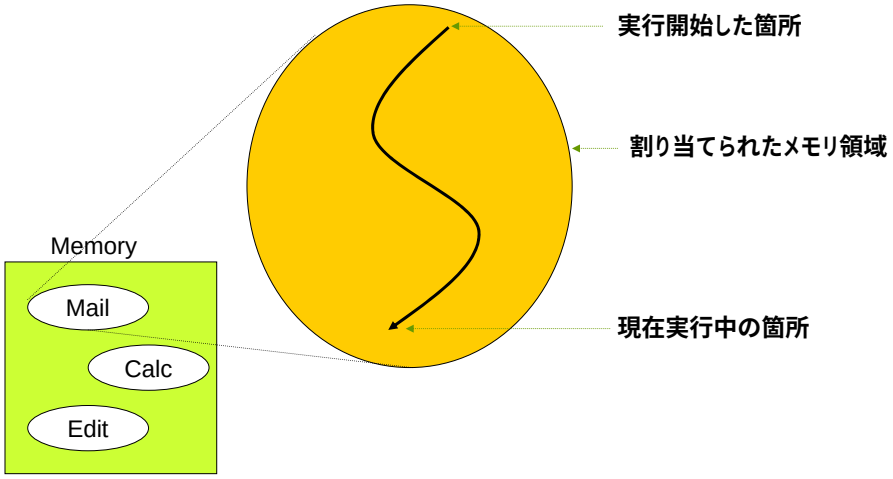
NAIST

アウトライン

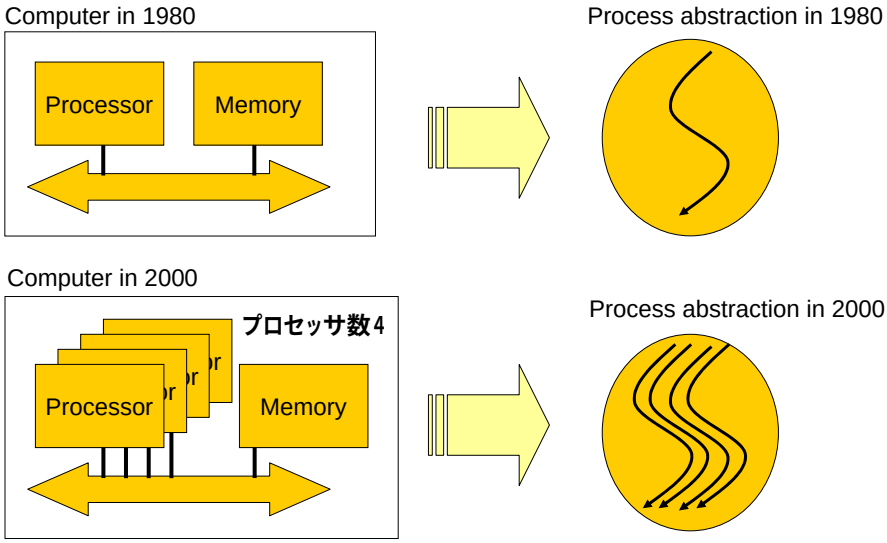
- プロセスとスレッド
- スレッドの管理
- スレッドの生成と抹消
- スレッドの状態
- スレッドの切り替え

- コンカレンシ
- セマフォ

プロセスのメンタルモデル

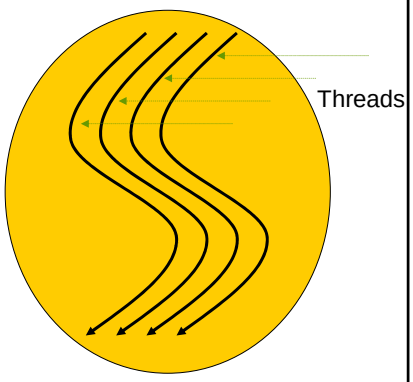


計算機の発達と抽象化



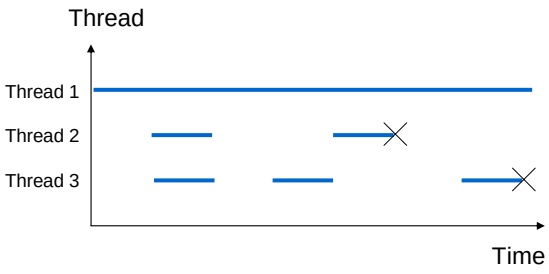
プロセスとスレッド

- プロセス：
  - メモリ割り当ての単位
- スレッド
  - 実行（プロセッサ割り当て）の単位
  - 並行処理を表現する1手法



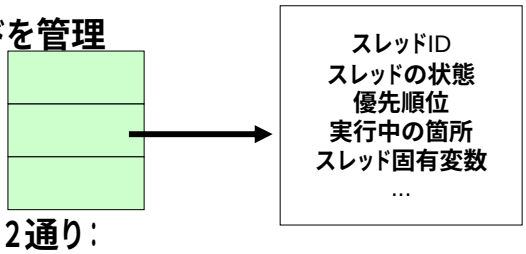
スレッドの生成と抹消

- プロセス開始時に1つ存在
- プログラム中、必要に応じてスレッドを生成
  - プロセッサ数を大きく超えてスレッドを生成しても性能が悪くなるだけなので要注意

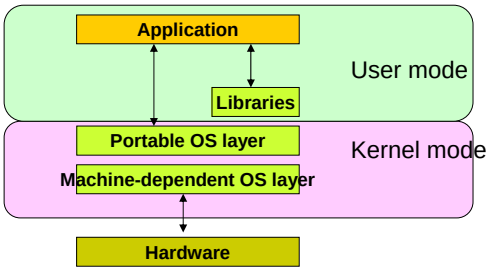


スレッドの管理

- プロセスごとにスレッドを管理



- スレッドの管理方法は2通り：
  - ユーザ空間（ライブラリ）
  - カーネル

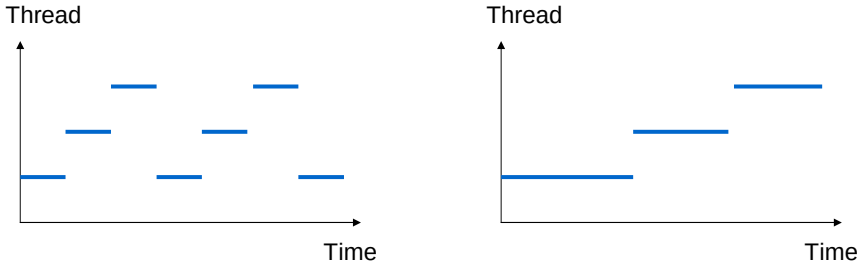


スレッドの状態

- 同一プロセス内のスレッドと状態を共有
- 各スレッド固有の状態：
  - 実行中の箇所（プログラムカウンタ）
  - スレッド固有変数
  - プロセッサの内部状態（レジスタ）

スレッドの切り替え

- スレッド数＞プロセッサ数の場合、プロセスと同様に時分割でプロセッサ時間を割り当てる必要がある
- Pre-emptive thread scheduling
- Non pre-emptive thread scheduling



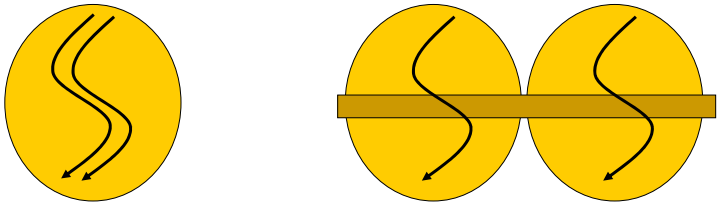
スレッドは必須か

安易なスレッド利用が招く3種のリスク

- 明示的なスレッドの生成、抹消
  - ハードウェア構成にあわないスレッド数
  - → 惨めな性能低下の危険性
- メモリ空間の共用
  - バグのあるスレッド同士でメモリ空間を共用
  - → 共倒れの危険性
- スレッド・スケジューラ
  - スケジューラの詳細を知らずにプログラム作成
  - → 飢餓状態、性能低下の危険性

スレッドか、プロセスか

- 単一プロセス、複数スレッドの代わりに、複数のプロセスを生成し、異なるプロセッサに割り付けてもよい
- プロセス間でメモリを共有することは可能



複数スレッド（プロセス）動作時の競合問題(1)

- deposit(N, x)
```
{
  a = read(N);
  a = a + x;
  write(N, a);
}
```

- withdraw(N, x)
```
{
  a = read(N);
  a = a – x;
  write(N, a);
}
```

銀行口座への預け入れと引き出しが、もし同時に実行されたら？
このような競合する可能性のある操作を正しく行うには？

複数スレッド（プロセス）動作時の競合問題(2)

- 哲学者の晩餐問題
- （TAによるデモ）

Semaphore

- P() または wait()
  - セマフォが正の整数になるまで待ち、1減算
- V() または post()
  - セマフォを1増やし、待っているプロセス（スレッド）を起こす
- （TAによるデモ）

競合問題のセマフォによる解決

- initialize(S, 1);

- deposit(N, x)
```
{
  wait(S);
  a = read(N);
  a = a + x;
  write(N, a);
  post(S);
}
```

- withdraw(N, x)
```
{
  wait(S);
  a = read(N);
  a = a – x;
  write(N, a);
  post(S);
}
```

Critical section

Mutual exclusion

コンカレンシの記述

- Semaphore
- Mutex
- Lock
- Monitor
- Message queue
- Condition variable
- ...

- (TAによるサンプルプログラム)

レポート

- **提出期限**：4/19 17:00 (**日本時間**)
- A**棟**3F **山口研究室入り口レポート提出ボックス**