コンピュータアーキテクチャー & オペレーティングシステムグループ

アーキテクチャーと オペレーティングシステム

❖コンピューターというハードウェアを音楽プレイヤーなどの 応用プログラムに「使いやすい形」で提供する「 基本的プログラム」がオペレーティングシステムです。「使いやすいかたち」に抽象化された コンピューターの構成をアーキテクチャーといいます。

現在の研究テーマ

❖現在卒論およびその準備として以下のテーマで研究を行なっています。 各テーマのパネルに詳しい説明があります

- ➢ サーバーアプリケーションの性能解析
- ➢ マルチコアシステムによる電力効率改善
- ➢ Unix システムの基礎 (ソース解析など)
- ➢ 帰納論理プログラムによる画像識別*

*ポルト大学との共同研究

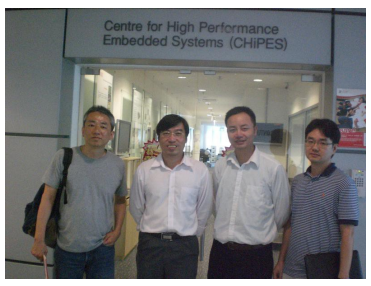
研究パートナー

❖"Politehnica" University of Timisoara (ルーマニア)

- ➢ 研究テーマ: リコンフィギュラブルコンピューティング

❖Nanyang Technological University (シンガポール)

- ➢ 研究テーマ:
  - 組み込みシステムとリアルタイム OS
  - マルチコアシステムアーキテクチャー
- ➢ 交換留学 (二星翔 2010 年 9 月から 12 月)
- ➢ 国際会議共催 (ISIC 2014)



❖Universidade do Porto (ポルトガル)

- ➢ 研究テーマ:
  - 論理プログラミング言語の並列化
  - 帰納論理プログラムによる画像識別
- ➢ 交換留学
  - 高橋和晃 (2009 年 8 月から 9 月)
  - Rui Rei (2010 年1月から4月, ポルトから)
  - 古河智弥 (2015年8月から9月)
- ➢ ポルト教員, 大学院生による特別講義

Computer Architecture & Operating Systems Group, The University of Aizu
Dr. Hitoshi Oi, Satoshi Ishizaka, Yasushi Nagao, Tsubasa Sugiyama, Tomoya Furukawa (Assoc. Member)

CAOS

コンピュータアーキテクチャー & オペレーティングシステムグループ

コンピューターと仮想化

❖コンピューターのハードウェアは複雑な構造で仕様も多種多様のため、そのままでは使いにくい。そこで色々な方法で「つかいやすく」見せかける工夫がされている。これを「仮想化」という。

マルチプログラミング

❖一台のコンピューターで常時複数のプログラムが走っている。例えば音楽プレイヤー，ネットワーク処理，画面処理等々。

❖マルチプログラミング環境では、それぞれのプログラムが単独で動いているように仮定してプログラムすることができる。

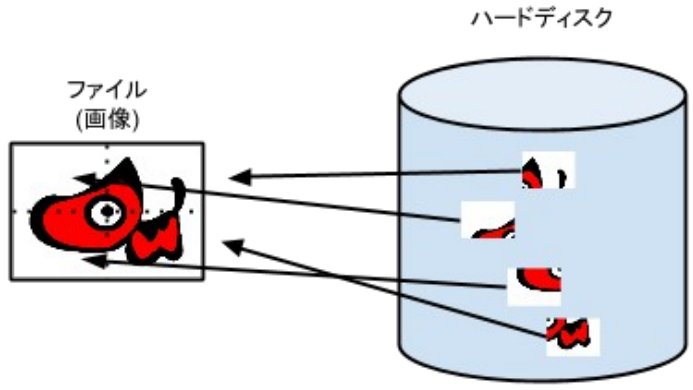
❖OS が実行中のプログラム (プロセス)を切り替え、各プロセスに独立した実行環境を提供する。また「入力待ち」のプロセスがシステム全体を止めたりせず、効率を上げるよう調整する。

ファイルシステム

❖コンピューターにはプログラムや多種のデータ(写真，文章，動画など)が大量に保存されている。

❖これらのデータはハードディスクなどに記録される。ハードディスクはたくさんの「番号がふられた一定の容量の箱」(ブロック)からできている。

❖人間や応用プログラムにとってはブロック番号でアクセスや管理するのは使い勝手が悪い。OS は各情報を「ファイル」という単位で管理し、「名前」「フォルダー」でのアクセスや、セキュリティ (管理者のみアクセスできる、等)を提供する。



ファイルシステムはディスクブロックとファイルの対応や、アクセス権限、空きブロックの管理などを行う。

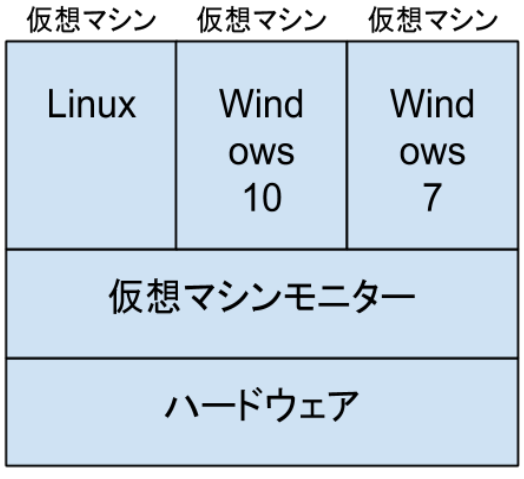
仮想マシン

❖一台のコンピューター (物理マシン)を複数の「独立したコンピューター」(仮想マシン)に見せる技術。

❖利点:

- 各マシンで別々の OS (例: Windows と Linux)を同時走らせる事ができる。
- コンピューター自体の価格に加え、ランニングコスト (電気代など)を抑える事ができる。

❖仮想マシンの OS による物理マシンのハードウェアへのアクセスにより、仮想マシン間での干渉が起きる。ハードウェアと仮想マシンの間に置かれた「仮想マシンモニター」(VMware など)がハードウェアへのアクセスを調整し、干渉を抑える。



Computer Architecture & Operating Systems Group, The University of Aizu
Dr. Hitoshi Oi, Satoshi Ishizaka, Yasushi Nagao, Tsubasa Sugiyama, Tomoya Furukawa (Assoc. Member)

University of Porto

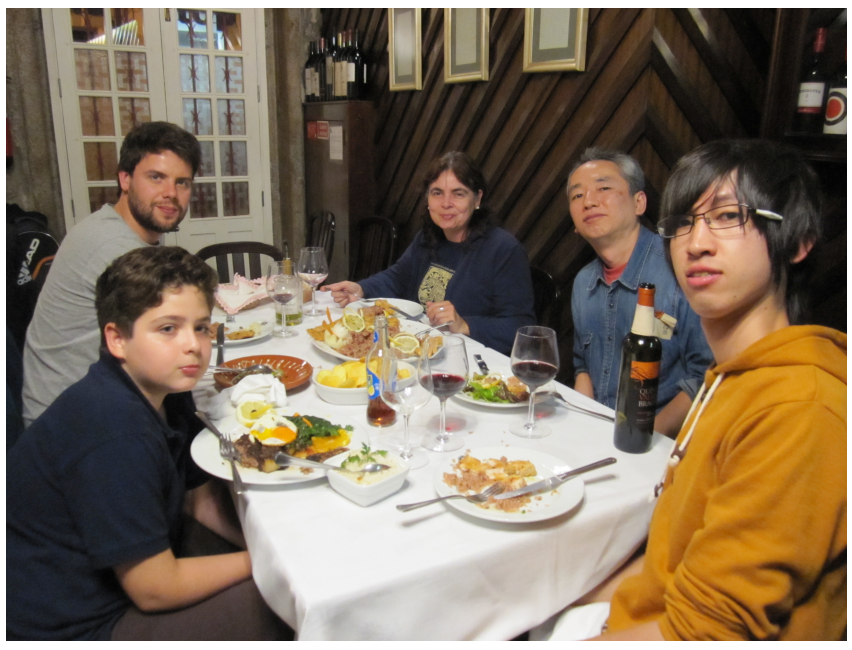
s1200191 Tomoya Furukawa

留学について

❖ポルトガルのポルト大学のコンピューターサイエンス学部へ留学し，研究を行った。期間は　2015/08/23　~ 2015/09/27 だった。

❖ポルトはポルトガル第二の都市で，ポルトガル北部の港湾都市。ポルトワインが有名。「リスボンは遊び，ブラガは祈り，ポルトは働く」という言い回しがあり，ポルトには会社が多い。

❖この留学は会津大学の海外派遣留学支援事業の支援で行った。



教授について

❖ポルトでは，Inês Dutra 教授 (テーブル奥左) の下で研究を行った。 Inês Dutra 教授はカンファレンスの準備など忙しい中質問や依頼に素早く対応して頂き，週一回の会議を開いてくださった。また，家族の誕生日パーティーに私を招待して下さるなど，大変良くして下さった。

環境について

❖気候について。会津若松の気温より涼しく，避暑地のような気候だった。

❖言語について。大学内ではほとんど英語で問題なかった。授業なども学生に合わせて英語を使用するそうだ。街中はホテルや観光客向けのレストラン以外では通じないこともあった。

❖食事について。ポルトガル料理には，魚も肉も米もあり，味付けも日本人にとって受けいられやすい味だと思った。アルコールはほぼ水とおなじ値段で昼から飲むのが普通とのことだった。

研究内容

テーマ:「帰納論理プログラミング」

❖Induction Logic Programming (ILP) とは，コンセプトラーニングとよばれる，機械学習を利用しあるデータの集合からある概念を導き出すということを，論理型プログラミング言語を使って行うもののことである。

❖概念というのは，例えば，「太郎の父は英雄である」，「太郎の母は花子である」というようなデータの集合から，「太郎の親は英雄または花子である」というようなデータの集合よりも一段階一般的な性質のことを言う。

❖このような学習を行う Aleph というようなシステムが Prolog で実装されており，留学中はそれを使用した。

Computer Architecture & Operating System Group, The University of Aizu
Dr. Hitoshi Oi, Tomoya Furukawa.

汎用OSとRTOSでの優先度の違いにおける処理時間の測定

s1200192 Ishizaka Satoshi

リアルオペレーティングシステム (RTOS)

❖時間的な制約がある処理を実行するための機能や特性を備えたオペレーティングシステムである。自動車や家電製品、携帯電話などの様々なデバイスに組み込まれている。

汎用OSとの違い

❖汎用OSは命令を実行するとき、ある順番にそって処理実行する。しかし車載機器のABS(アンチロックブレーキシステム)のような大きな事故や被害につながるような命令は即座に実行されなければいけない。このようにRTOSは定められた時間内に処理が完了されることを保証する必要がある。

実験内容

❖Ubuntuベースのマシン上にそれぞれLinuxカーネル4.2.0 (非RTカーネル)と3.18.21-rt (RTカーネル) による仮想マシン2つを起動。

❖負荷をかけた状態で割り込み処理を発生させその処理にどの位時間がかかるのかを測定した。

使用ツール

❖・hackbench

➢複数のプロセスを自動で生成しパイプを用いたI/O処理を頻繁に行う。（デフォルトで40個のプロセスを生成)

❖・cyclictest

➢周期的に割り込み処理を発生させる。

➢優先度を自由に変更できる。

手順

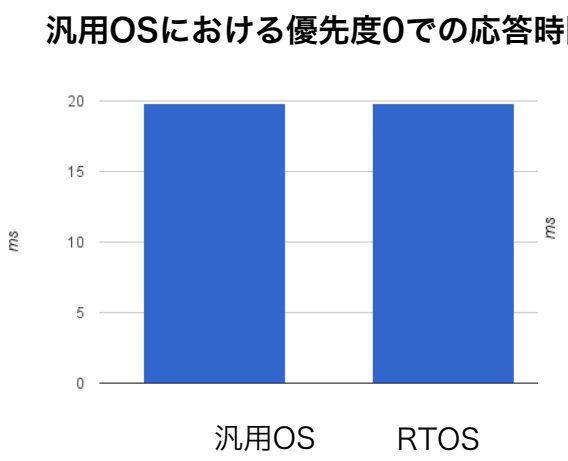
❖それぞれのカーネルにおいて優先度0と優先度99で割り込みを行う。

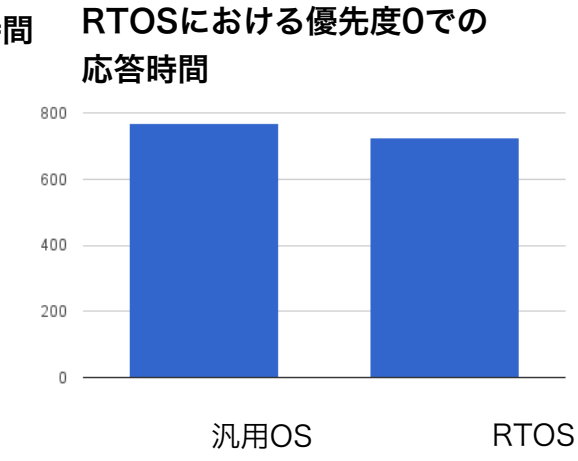
➢./hackbench 100(100*40 = 4000プロセス)

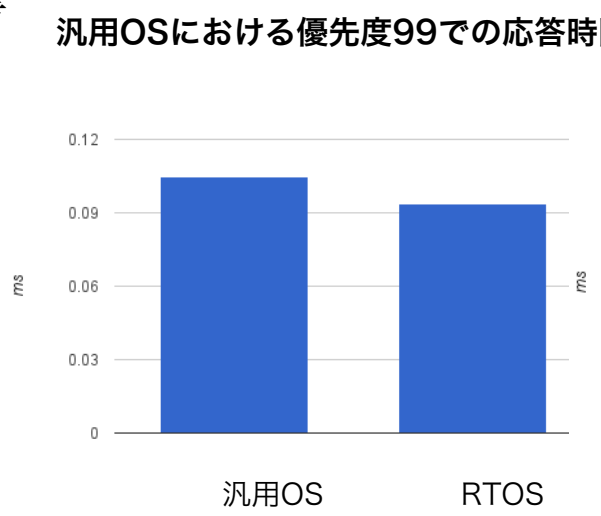
➢cyclictest -t 1 -p 99 -i 5000 -l 500(優先度0の時)

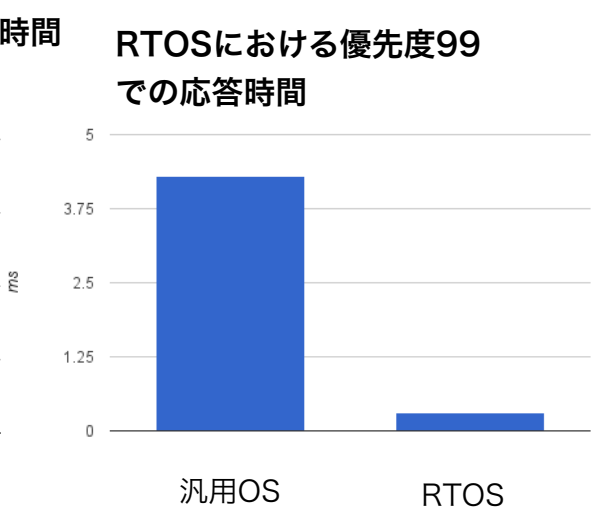
➢cyclictest -t 1 -p 99 -i 5000 -l 500(優先度99の時)

結果









考察

❖それぞれのカーネルにおいて優先度が0の時はほぼ変わらない結果となったが、優先度を99に上げると圧倒的にRTカーネルの方が処理に時間がかからなくなったのがわかる。

❖また、非RTカーネルだと平均と最悪の場合で差が大きいがRTカーネルだとその差が小さいのがわかる。これはRTカーネルが応答時間を保証しているためだということがわかる。

今後

❖VCPUの数や負荷のかけ方、割り込み処理の周期など条件を変えて試行回数を増やして行きたい。また、仮想上のVCPUの動作よって実CPUがどのような動きをするのかを見ていきたい。

Unix/Linux コマンドプログラミング

s1210241 学部３年　Yasushi Nagao

Unixについて

- Unixとは1969年にAT&T社ベル研究所で開発が始まったオペレーティングシステムである。
- 異なる機種間での移植性や、複数プログラムを並列して動作させられるマルチタスク、複数の利用者が一つのシステムを利用できるマルチユーザなどを重視して設計された。
- コマンドライン上で様々なコマンド(プログラム)を実行することでコンピュータの操作のすべてがコマンドライン上で行える。

前期の学習テーマ

- Unixコマンド(プログラム)がどのように実装されていて、どのように動いているのかを実物を見ながら学ぶ。例題として以下を説明する。
  - lsコマンド
  - whoコマンド

ls

- ls(list segments)はディレクトリ(フォルダ)の中身をリスト表示するコマンドである。通常は現在のディレクトリ(カレントディレクトリ)の中身をリスト表示する。
- lsを実現するには、プログラム側でディレクトリを開き、ディレクトリの内容を読み出し、出力する必要がある。
- これらの動作をするシステムコールがC言語に用意されている。それを用い、opendirでディレクトリを開き、要素数がなくなるまでreaddirで内容を取得し、closedirでディレクトリを閉じる。最後に内容を出力することでlsコマンドが出来上がった。

who

- 現在ログインしているユーザの情報を表示する。

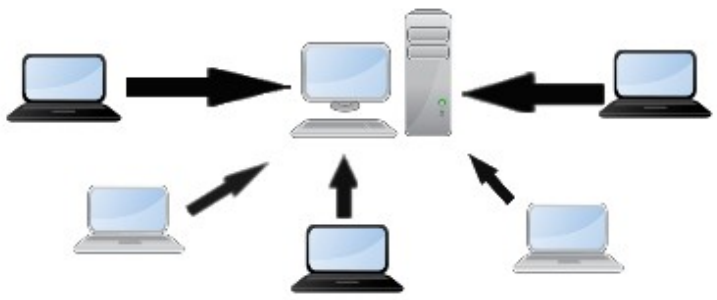


図 1:ログイン接続の状況

- 通常、ログイン情報は/var/adm/utmp(varディレクトリの内のadmディレクトリの中のutmpファイル)に記録されている。whoはそれを開き、出力すればよい。
- utmpファイルにはログイン情報が構造体として記録されている。whoコマンドを実現するには、プログラム側で/var/adm/utmpを開き、utmpファイルから構造体を取得し、出力する必要がある。
- lsと同様にC言語のシステムコールを用い、openでutmpファイルを開き、readでutmpの構造体の情報(1つの構造体が1ユーザのログイン情報になっている)をすべて取得し、closeでutmpファイルを閉じる。最後に情報を出力することでwhoコマンドが出来上がった。

現在の実験(学習)

- 現在のプロセッサーの特徴などの学習のためSimpleScalarというアーキテクチャシミュレーターを用いた実験を行っている。
- スーパースカラー(命令のフェッチ数やALUの数)、キャッシュ(容量やアルゴリズム、連想度)、分岐予測などの設定を変化させ、同じプログラムを実行することで、ハードウェアがソフトウェアの実行にどのような影響を与えるのかを見ながら学習を進めている。

参考文献

[1]Unix/Linux プログラミング理論と実践 (著)Bruce Molay (訳)長尾 高弘

UNIX V6

学部 3 年 s1210052 Tsubasa Sugiyama

UNIX V6

❖とは Kenneth Lane Thomson と Dennis MacAlistait Ritchie により開発され、1975 年にベル研究所からリリースされた PDP11 というプロセッサ上で動く OS です。UNIX V6 は高級言語である C 言語で書かれており、かつ、ソースコードが公開されていたため、大学などはこれを入手し自分の環境に移植していき、UNIX V6 は広く使われていきました。

UNIX の歴史

❖ベル研究所は UNIX V6 のリリース後、1979 年に UNIX V6 を基にした最初の BSD のリリースしました。その後、それぞれにバージョンアップ版や派生版がリリースされていきました。のちに、標準化の動きがあり、POSIX と呼ばれる OS の共有 API 規格が策定されました。かの Linux も POSIX 準拠を目指して開発されています。つまり最新の OS のほとんどが、基をたどると UNIX(V6)に行き着きます。現在スマートフォンとして、よく知られている Andoroid も LINUX ベースのシステムなので、祖先は UNIX V6 ということになります。

なぜ UNIX V6 を学ぶのか

❖なぜ最新 OS ではなく、少し古い UNIX V6 を学ぶのかというと、UNIX V6 は現在使われている 1000 万行を超えるソースコードからなる最新 OS の祖先であり、約一万行という簡潔なソースコードの中に OS の基本となるアイディアがほとんど実装されています。さらにロジックがシンプルで、システム全体を追うことができるという点において最新の OS の理解の手助けになります。

実際にエミュレーター上で動いている UNIX V6

❖ 実際に PDP11 エミュレーター上で UNIX V6 が動作している様子

```
# ls
bin
dev
etc
lib
mnt
mnt2
rkunix
rkunix.40
tmp
unix
usr
usr2
```

➢ これが UNIX V6 の全体であり、上の画像から構造のシンプルさがわかります。

現在の勉強内容

➢ 現在は、現代プロセッサの機能である、スーパースカラやキャッシュや分岐予測について学んでいます。SimpleScalar というマイクロアーキテクチャのシュミレーターを使い、先ほど言った機能をオプションで変更し、キャッシュミス率などの値がどのように変わるか見ながら CPU の動作に対する理解と勉強をしています。

➢ 参考文献

[1]はじめての OS コードリーディング　UNIX V6 で学ぶカーネルの仕組み (著)青柳　隆宏