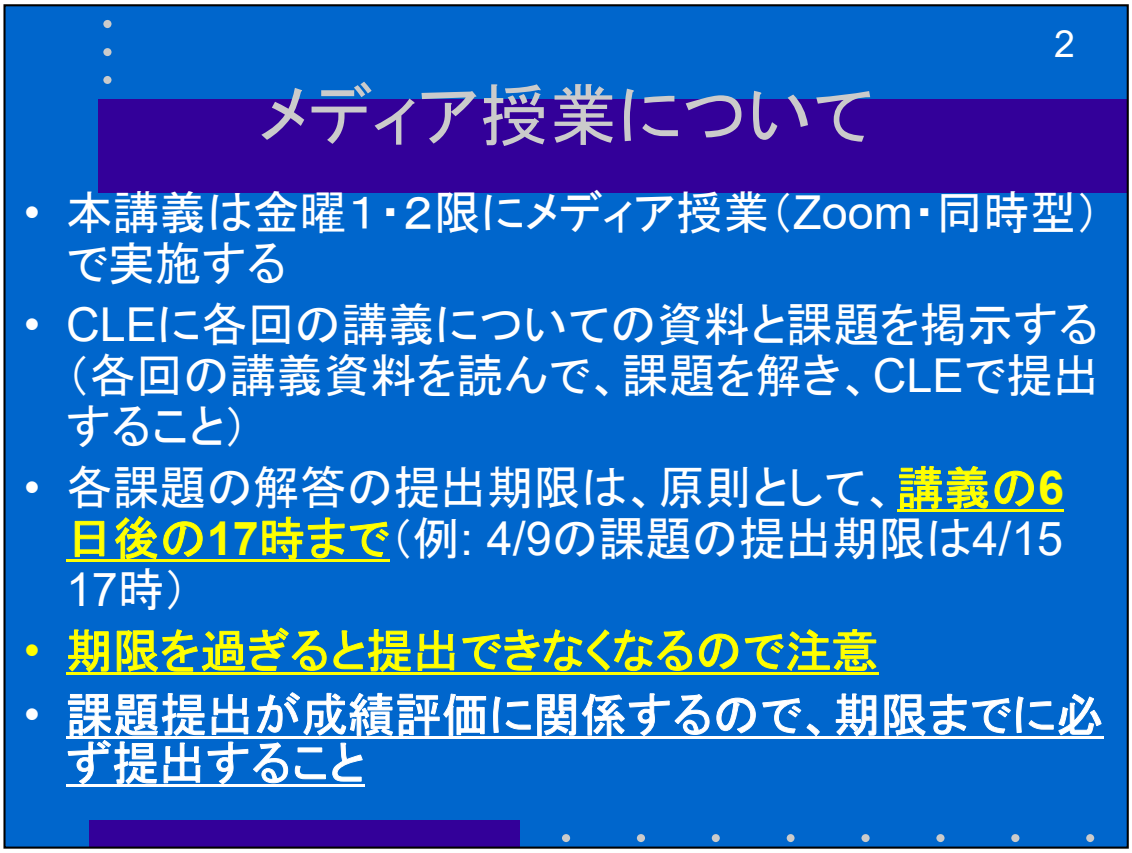
# オペレーティングシステム

資料　第 1 分冊(2021)

村田正幸　(murata@ist.osaka-u.ac.jp)
○松田秀雄(matsuda@ist.osaka-u.ac.jp)




本講義は金曜1・2限にメディア授業（Zoom・同時型）で実施します。

各講義ごとに課題を課すので、提出期限（原則として、講義の6日後の17時）までに必ずCLEで解答を提出すること。

期限を過ぎると解答を提出できなくなるので注意。

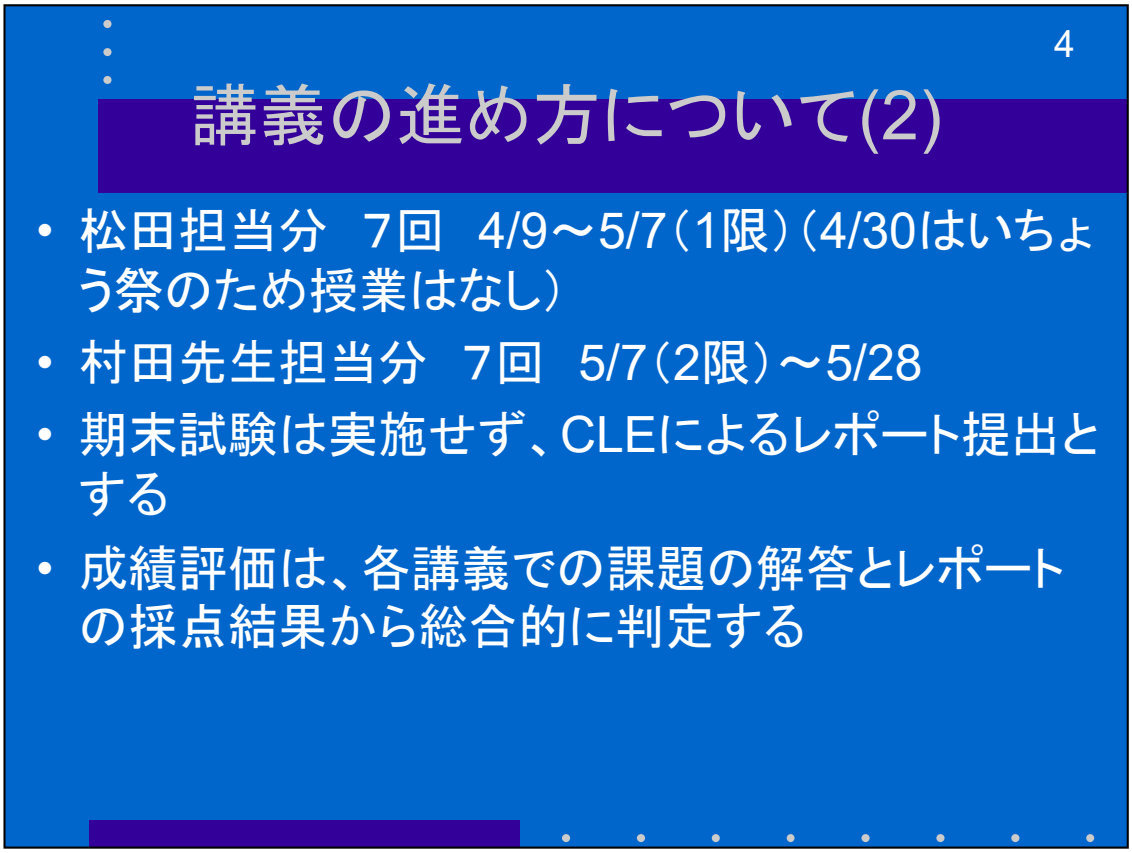
また、後述するように、**課題提出が成績評価に関係するので、期限までに必ず提出すること。**

3

# 講義の進め方について(1)

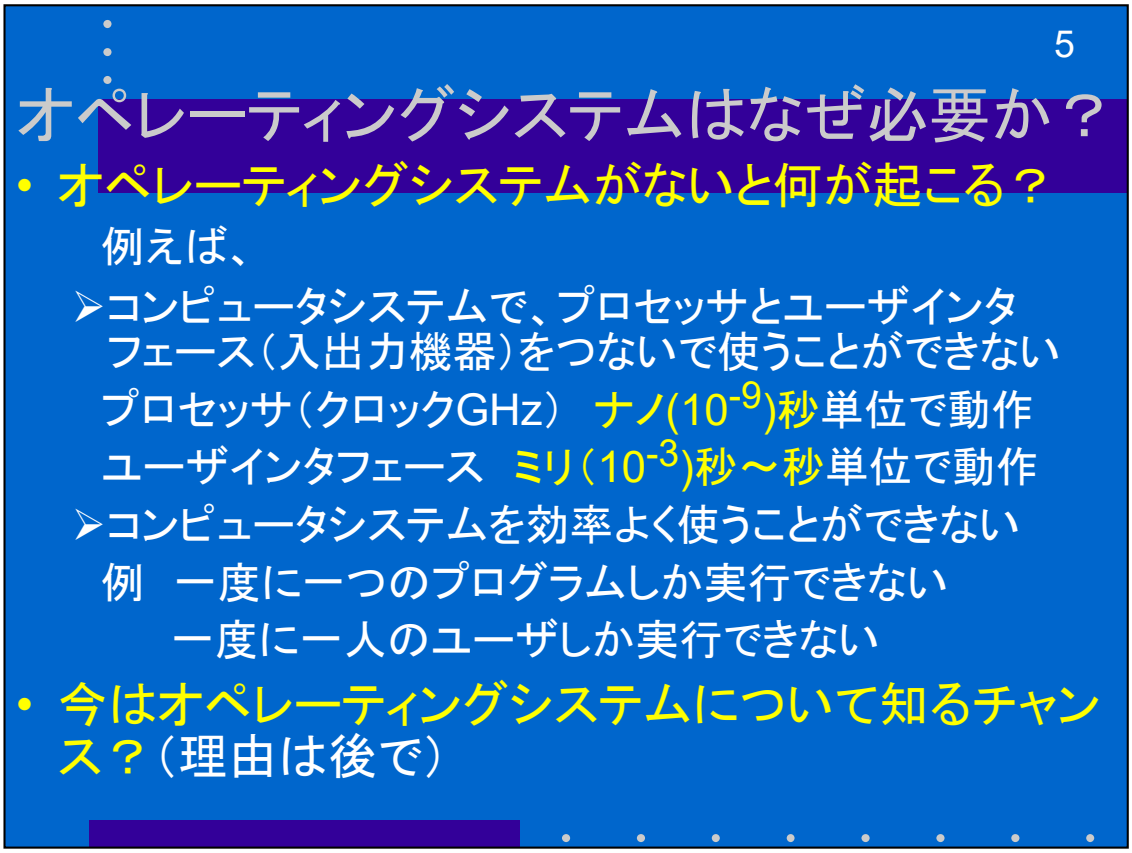
- 情報科学科 計算機・ソフト　3年次配当（必修）
- 「計算機アーキテクチャ」の受講を前提とする
- 教科書　近代科学社 「コンピュータサイエンスで学ぶ　オペレーティングシステム　OS学」
- ほぼ教科書に沿って講義を進める
- 教科書にとりあげられていない話題については、別途説明する
  - ➢デッドロックと資源割り当てなど

この講義は、必修科目です。
教科書に沿って講義をするので、教科書を用意してください。




前半7回と後半7回で担当が変わります。

期末試験は、CLEによるレポート提出で替えることがあります。




オペレーティングシステムには2つの役割があります。

1つ目は、プロセッサとユーザインタフェースとの間の動作時間の差を埋めることです。

プロセッサは、ユーザインタフェースと比べてはるかに速く動作しているので、これらの差を埋める必要があります。

2つ目は、コンピュータシステムを効率よく使うことです。

# オペレーティングシステムについて知る必要性(1)

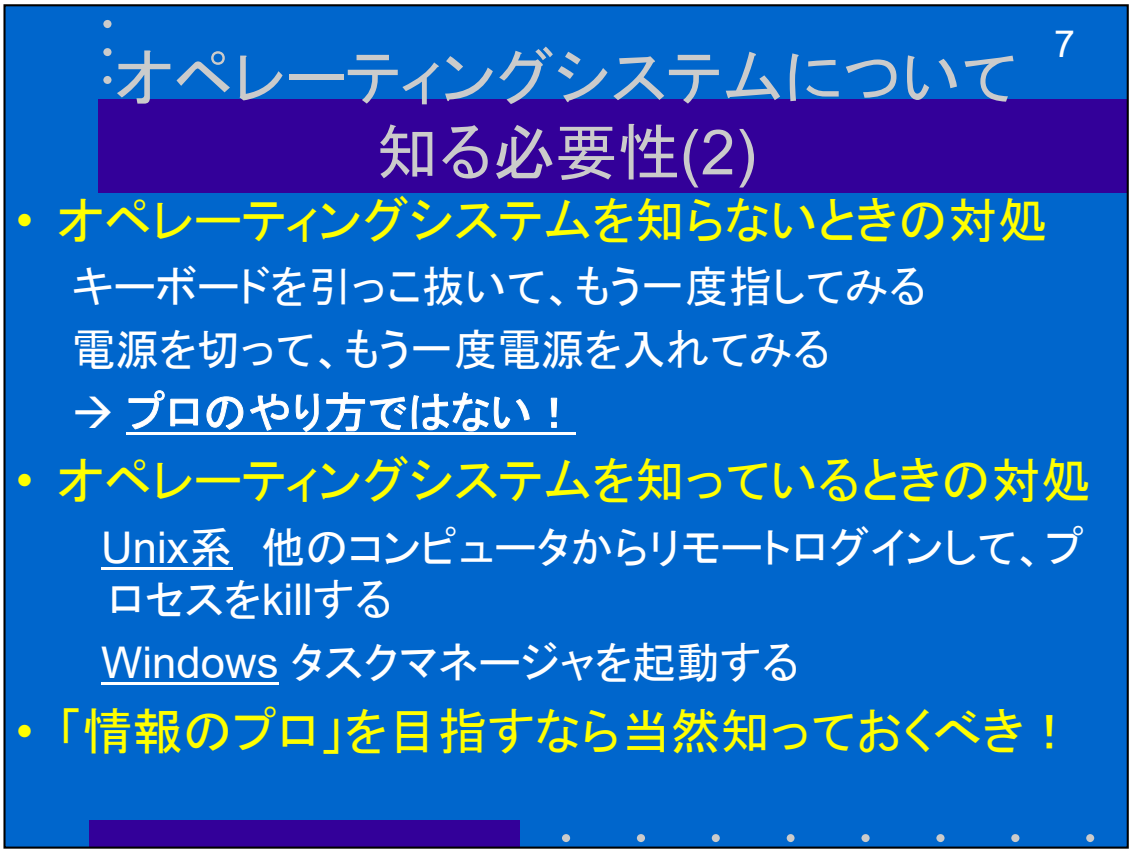


- オペレーティングシステムを知らないと何が起こるか？
  - コンピュータシステムの異常時に、ハードウェアの障害かソフトウェアの障害かを切り分けできない

    例　キー入力をしても何も反応しなくなった

オペレーティングシステムについて知らないと、コンピュータシステムに何らかの異常が起こった時に、その原因がハードウェアによるものなのか、ソフトウェアによるものなのかを切り分けることができません。

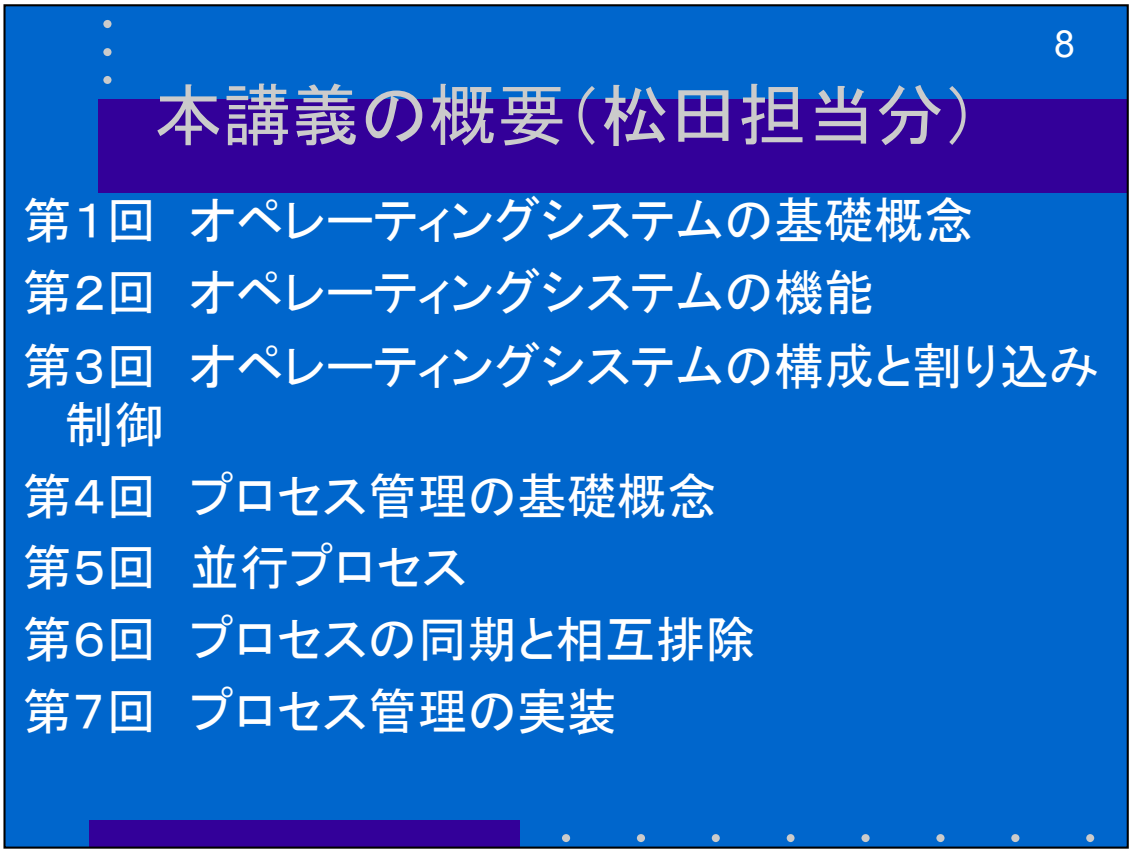
一例として、突然、キーボードから何かキーを入力しても、何も反応しなくなったときのことを考えてみましょう。



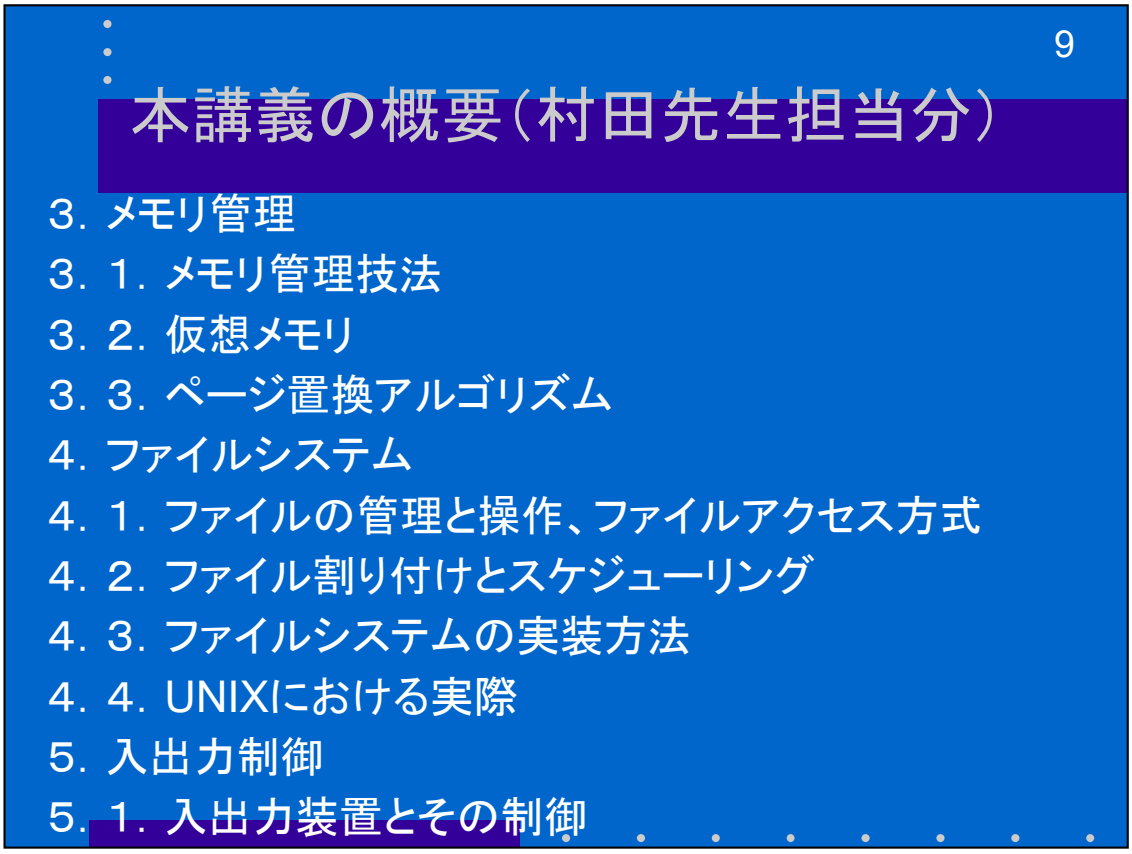

キーボード入力に何も反応しなくなったからといって、いきなりキーボードを引っこ抜いてみたり、電源を切るのは、情報科学科の学生のような専門知識を持つ人のするやり方ではありません。

オペレーティングシステムについての知識があれば、例えば、Unix系のOSであれば、別のコンピュータからリモートログインして、原因となりそうなプロセスがあればこれを強制終了させたり、Windows系のOSであればタスクマネージャを起動して同様のことを行うなどで解決する可能性があります。

このように、オペレーティングシステムについての知識があれば、他への影響が少ないような対処方法を取ることができます。




松田担当分の各回の講義内容を示しています。




村田先生担当分の講義の概要です。

# 1.1 OSの基本的な役割
## ー 現代のOSと各世代のOS ー

10

- オペレーティングシステム： Operating Systemの頭文字を取って**OS**と書くことが多い
- OSの位置付け
  - ➢OSはわかりにくい?
  - ➢ソフトウェアの中では、一番、ユーザからなじみのない位置にある
  - ➢直感的な説明:「OSは、コンピュータシステムを構成するソフトウェアの中で、ユーザプログラムと言語処理プログラムを除いたもの」
  - ➢これらを除いた後、いったい何が残っているのか？

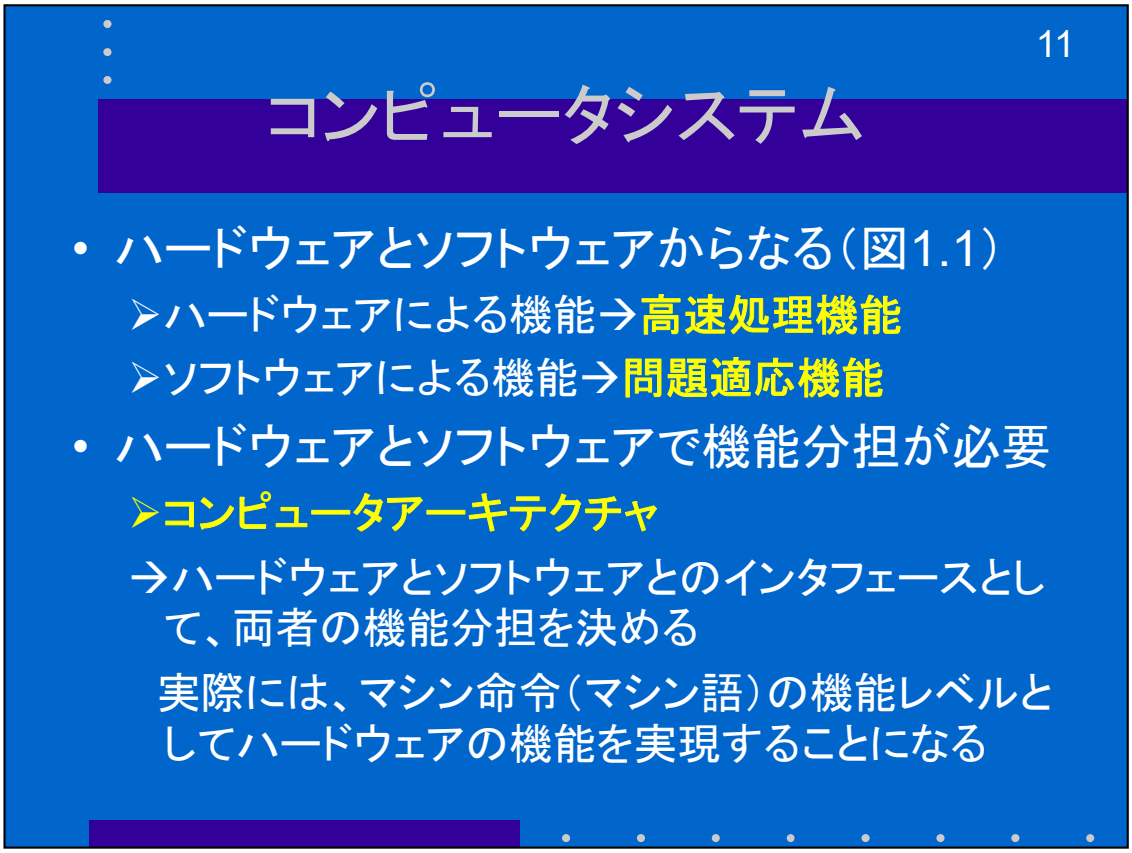
オペレーティングシステムは、英語の頭文字を取ったOSという略称で呼ばれること多いです。

OSは、コンピュータシステムを構成するソフトウェアのうちの一つですが、ユーザには一番なじみのない位置にあります。

これについては、以降で説明します。

OSの直感的な説明は、「コンピュータシステムを構成するソフトウェアの中で、ユーザプログラムと言語処理プログラムを除いたもの」というものです。

ユーザプログラムと言語処理プログラムを除くといったい何が残るのでしょうか？




コンピュータシステムは、ハードウェアとソフトウェアから構成されていますが、それぞれの役割が異なります。

このスライドで、図1.1と書かれていますが、これは教科書の中の図1.1を参照しているということを表しています。

以降のスライドで図〇〇という表示があった場合も同様ですので、教科書を用意するようにしてください。

ハードウェアは、主にシステムの高速処理機能を役割としています。

一方で、ソフトウェアは、主に問題適応機能を役割としています。

このため、コンピュータシステムの機能のうち、どの部分をハードウェアで実現し、どの部分をソフトウェアで実現するかという役割分担を考えることが必要で、これを決めるのはコンピュータシステムの設計、すなわち、コンピュータアーキテクチャで考えることになります。

もう少し具体的に言うと、ハードウェアとソフトウェアとのインタフェースを考える中で、両者の機能分担を決めていくことになります。

これは実際には、プロセッサのマシン命令（マシン語）にどれだけのレベルの機能を実現することになります。

# コンピュータシステムの機能の実現

12

- **ハードウェアでの機能の実現**
  → マシン命令の機能を作成（既存のマシン命令の改善が主流）
- **ソフトウェアでの機能の実現**
  → 言語処理プログラム（コンパイラなど）によって、プログラムをマシン命令の列に翻訳または変換する（図1.2 コンピュータシステムによる情報処理過程）

コンピュータシステムの機能を実現する上で、ハードウェアとソフトウェアがどのように関係するかを、もう少し見ていきましょう。

ハードウェアでの機能の実現は、前のスライドで書いたように、マシン命令の機能を作成により行うことになります。

これは、既存のプロセッサのマシン命令を基に、それを改善する方法が主流です。

一方、ソフトウェアでの機能の実現は、コンパイラなどの現処理プログラムによって、プログラムをマシン命令の列に翻訳または変換することになります。

教科書の図1.2にその処理過程が示されています。

## 基本ソフトウェア(= システムプログラム)と応用ソフトウェア(= ユーザプログラム)

13

- 基本ソフトウェア
  - コンピュータシステムに、原則として“唯一”登載されている
  - 基本プログラム、システムソフトウェア、システムプログラムともいう(本講義では「システムプログラム」と呼ぶ)
- 応用ソフトウェア
  - ユーザが後からインストールして使用する
  - 応用プログラム、ユーザソフトウェア、ユーザプログラムともいう(本講義では「ユーザプログラム」と呼ぶ)

コンピュータシステムのソフトウェアは基本ソフトウェアと応用ソフトウェアに分類できます。

基本ソフトウェアは、コンピュータシステムに原則として「唯一」つまり一つだけ搭載されています。

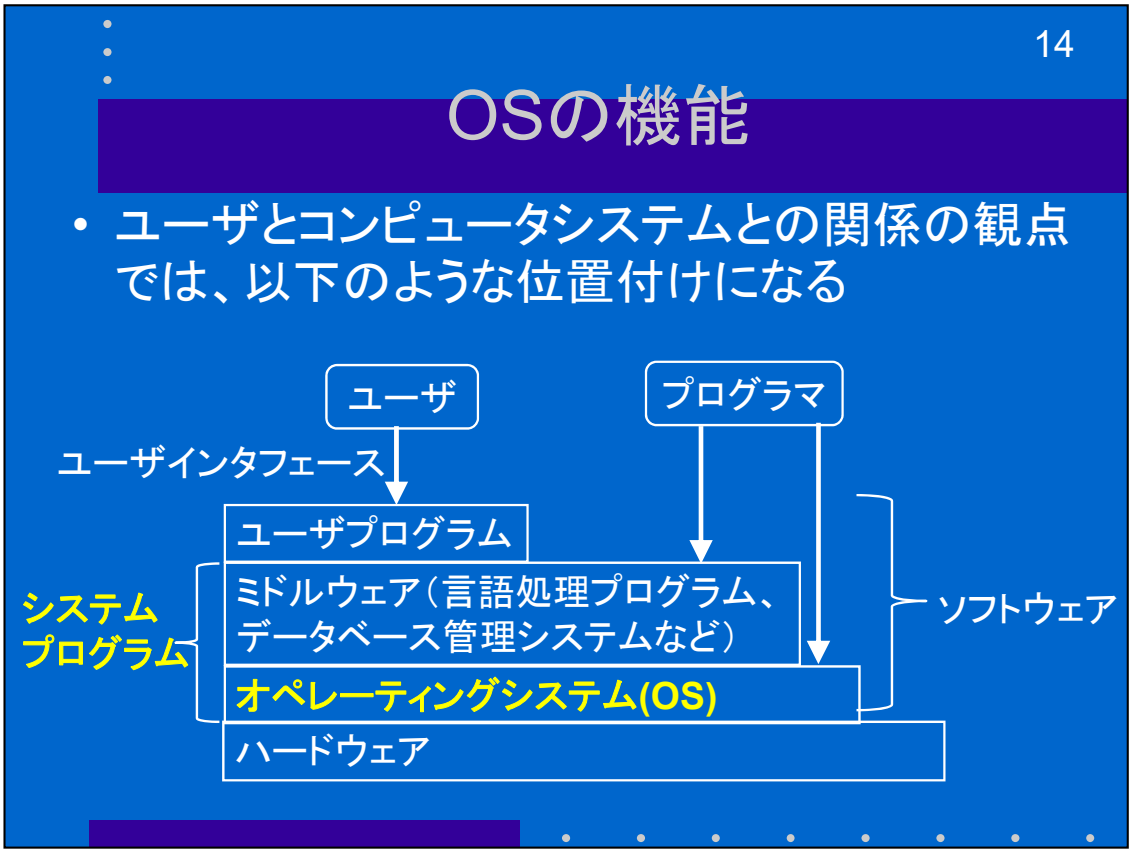
基本ソフトウェアは、基本プログラム、システムソフトウェア、システムプログラムとも呼ばれることがあります。

本講義では、この基本ソフトウェアは、以降、システムプログラムと呼ぶことにします。

一方、応用ソフトウェアは、基本的にはコンピュータシステムにインストールされておらず、ユーザが後からインストールして使用することになります。

応用ソフトウェアも、応用プログラム、ユーザソフトウェア、ユーザプログラムといろいろな名前で呼ばれます・

本講義では、応用ソフトウェアを、以降、ユーザプログラムと呼ぶことにします。




以上述べたソフトウェアの関係を、ユーザやハードウェアとの関係も含めて図で表すと、この図のようになります。

システムプログラムは、ソフトウェアの中ではハードウェア寄りに位置し、ユーザプログラムはユーザ側つまりユーザインタフェースを備えています。

オペレーティングシステムは、システムプログラムの中の一つであり、システムプログラムから言語処理プログラムなどミドルウェアを除いたものを指します。

ちなみに、ここでミドルウェアとは、ユーザプログラムとオペレーティングシステムの中間に位置することからこのように呼ばれることがあります。

なお、ユーザインタフェースの機能はユーザプログラムだけから提供されるのみではなく、ミドルウェアや、オペレーティングシステムもそれぞれのユーザインタフェースの機能を持っています（これについては、次のスライドで説明します）。

15

# OSの機能（つづき）

- OSは、システムプログラムの一部
- OSの機能は、次の2種類に分類できる
  1. 狭義のOS（OSの基本機能）
  2. **ユーザインタフェース**（OSの応用機能）
- 狭義のOSの機能の例
  - ハードウェア（プロセッサ、メモリ、入出力装置）の管理
- OSが提供するユーザインタフェースの例
  - コマンド
  - GUI（グラフィック・ユーザインタフェース）

前のスライドの図にあったように、OS（オペレーティングシステム）は、システムプログラムの一部です。

OSの機能は次の2種類に分類できます。

1つ目は、OSの基本的な機能で、これは狭義、すなわち、狭い意味でのOSと呼ばれるものです。

2つ目は、OSが提供するユーザインタフェースで、これはOSの応用機能と呼ばれることがあります。

狭義のOSの機能の例としては、ハードウェアの管理があります。

また、OSが提供するユーザインタフェースの例としては、Unix系のOSで言えばつシェルのコマンドや、GUI（グラフィックユーザインタフェース）があります。

# OSの発展史

16

- OSはコンピュータと共に発展
  →コンピュータの世代に対応してOSにも世代がある

| 世代 | コンピュータ | OS |
|---|---|---|
| 第1世代 | 真空管 | システムプログラムなし |
| 第2世代 | トランジスタ | 簡易システムプログラム |
| 第3世代 | IC（集積回路） | 本格的システムプログラム |
| 第4世代 | マイクロプロセッサ | 統合プログラミング環境 |
| 第5世代 | インターネット | 隠ぺいされた（標準化・共通化）システムプログラム |
| 第6世代 | ユビキタス | 透明なシステムプログラム |

コンピュータが発展してきた過程はいくつかのステップに分かれており、これを世代と呼びます。

OSはコンピュータと共に発展してきたため、コンピュータの世代に対応して、OSにも世代があります。

この表は、コンピュータの世代に対応して、OSにどのような世代があるかをまとめています。

# OSの発展史（つづき）

17

- OSの発展の歴史は繰り返す？

（注）以下の各行はコンピュータの世代とは無関係

| 携帯電話 | OS |
|---|---|
| 「携帯」する電話 | システムプログラムなし |
| 簡易ネットワーク機能 | 簡易システムプログラム（i-モード, EZweb, …） |
| スマートフォン | 本格的システムプログラム（iOS, Android, …） |
| クラウドサービス | インターネット上でサービスの提供（Google Apps, Amazon AWS… ） |
| ？ | ？ |

前のスライドでは、コンピュータの世代について説明しましたが、今では第1世代、第2世代、第3世代のコンピュータは目にする機会がないため、イメージがわきにくいかも知れません。

そこで、コンピュータではなく、携帯電話の発展との対応についてまとめたのがこの表です。

携帯電話は、もともとは、まさに携帯、つまり持ち運びできる電話であり、その機能は電話をかけるのみで、ほとんどハードウェアだけで実現されており、システムプログラムと呼べるものはほとんどありませんでした。

その次に、ネットワークに接続して情報検索などの機能を備えた携帯電話が登場しました。今ではガラケーと呼ばれる携帯電話がそれです。このような携帯電話のソフトウェアは、OSで言えば簡易システムプログラムに対応します。

その後さらに、スマートフォンと呼ばれるものが登場しました。スマートフォンの持つソフトウェアは、OSで言えば本格的システムプログラムに対応します。ちなみにスマートフォンの基本的なソフトウェアはiOSのようにOSと呼ばれており、そこで提供される機能はまさにコンピュータのOSと同じレベルのものです。

さらに今では、コンピュータでも提供されているクラウドなど、インターネット上でのサービスが提供されるようになっています。

このように、携帯電話は、コンピュータの発展と同じような発展のステップで発展しましたが、それぞれのステップで使われたソフトウェアは、まさにコンピュータの世代に対応したOSと同じような発展をしてきたと見ることができます。

18

# 第1世代（1940～1950）

- システムプログラムがない世代
- コンピュータシステムを、一人のユーザや単一のプログラムが占有して利用
- OSはない

- *通話機能しかない、「携帯」する電話に相当*

それでは、コンピュータの世代に対応するOSの機能を詳しく見て行きましょう。

まず、第1世代のコンピュータですが、これは前のスライドにもあったように、通話機能しかない、「携帯」する電話に相当します。

この世代では、システムプログラムと呼ばれるものは存在しておらず、コンピュータシステムは、一人のユーザが、単一のプログラムのみを実行することだけしかできませんでした。

OSと呼べるものはありませんでした。

19

# 第2世代（1950～1960）

- 簡易システムプログラム世代
- 簡単な割り込み処理プログラムを、システムプログラムとして実装
- プログラミング言語の開発と使用（言語処理プログラムの登場）
- バッチ処理（複数人のユーザが作成した複数個のユーザプログラムを一括して実行）が主流

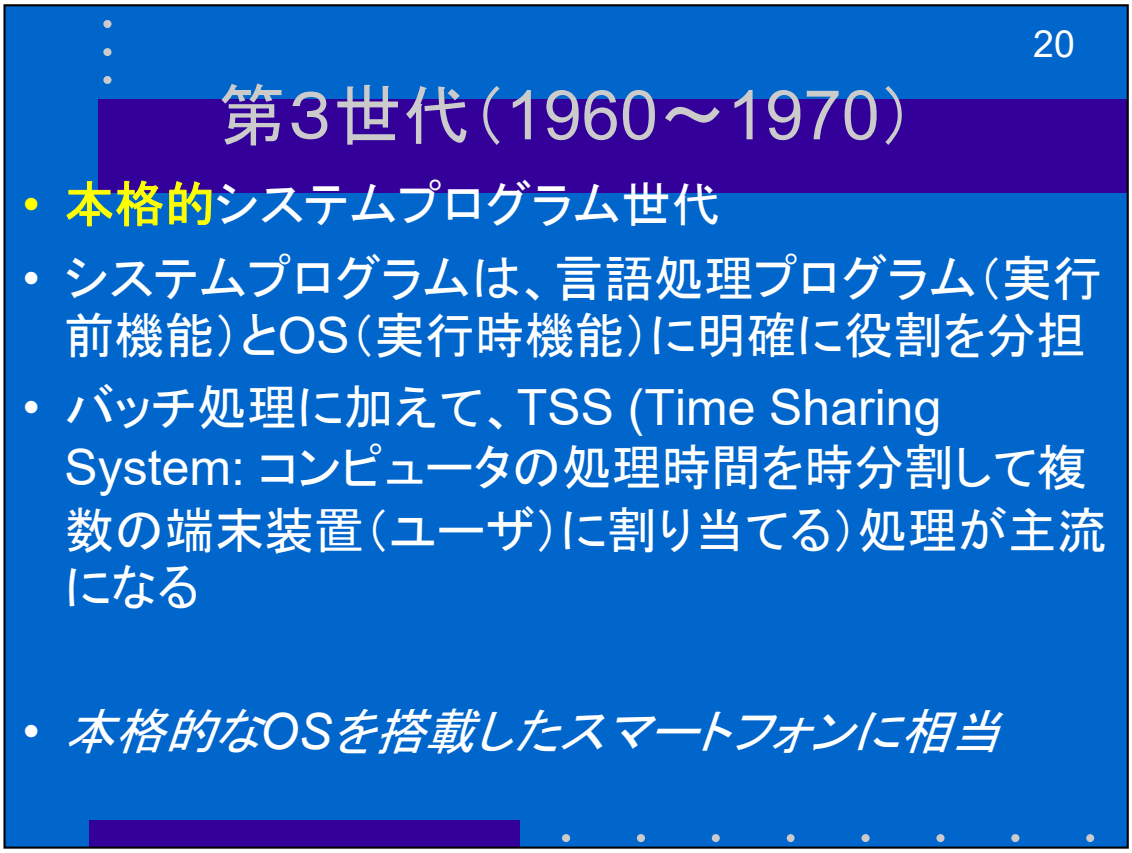
- *簡易ネットワーク(i-mode)機能のレベル*

第2世代のコンピュータは、携帯電話で言えば簡易ネットワーク機能のみを持つレベルに相当するものです。

そこでは、簡易システムプログラムと呼ばれるものが動作しており、簡単な割込み処理プログラムが実装されていました。

プログラミング言語が開発され、コンパイラなどの言語処理プログラムが登場して使われるようになった世代でもあります。

この世代では、複数人のユーザが作成した、複数個のユーザプログラムを一括して実行する、バッチ処理と呼ばれる方式が利用の主流でした。

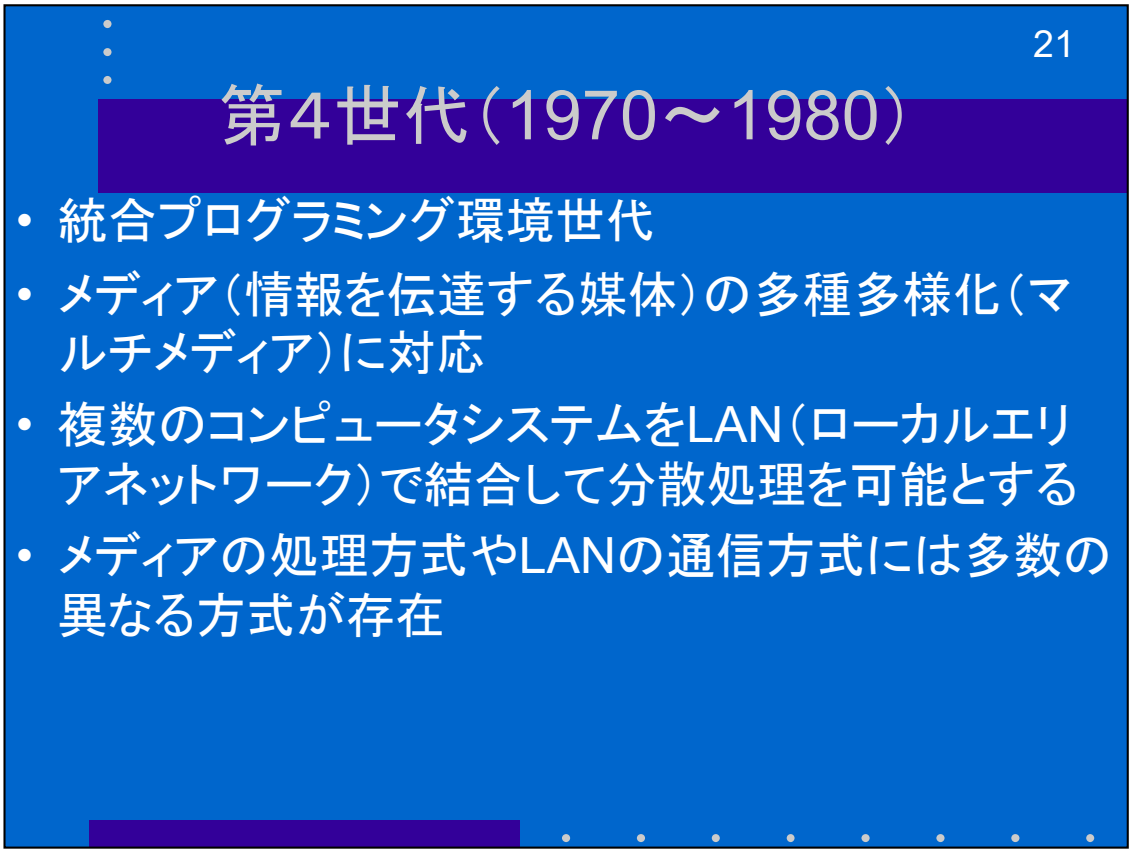



第3世代のコンピュータは、初めて登場したときのiPhoneなどのスマートフォンに相当します。

そこでは、本格的システムプログラムと呼ばれるソフトウェアが搭載されており、言語処理プログラムとははっきりと区別できるOSと呼べるものが現れました。

言語処理プログラムは、プログラムの実行前に使われ、OSはプログラムの実行時に使われる機能を果たします。

そして、OSにより、バッチ処理に加えて、TSS（time sharing system）と呼ばれる処理が主に使われるようになりました。

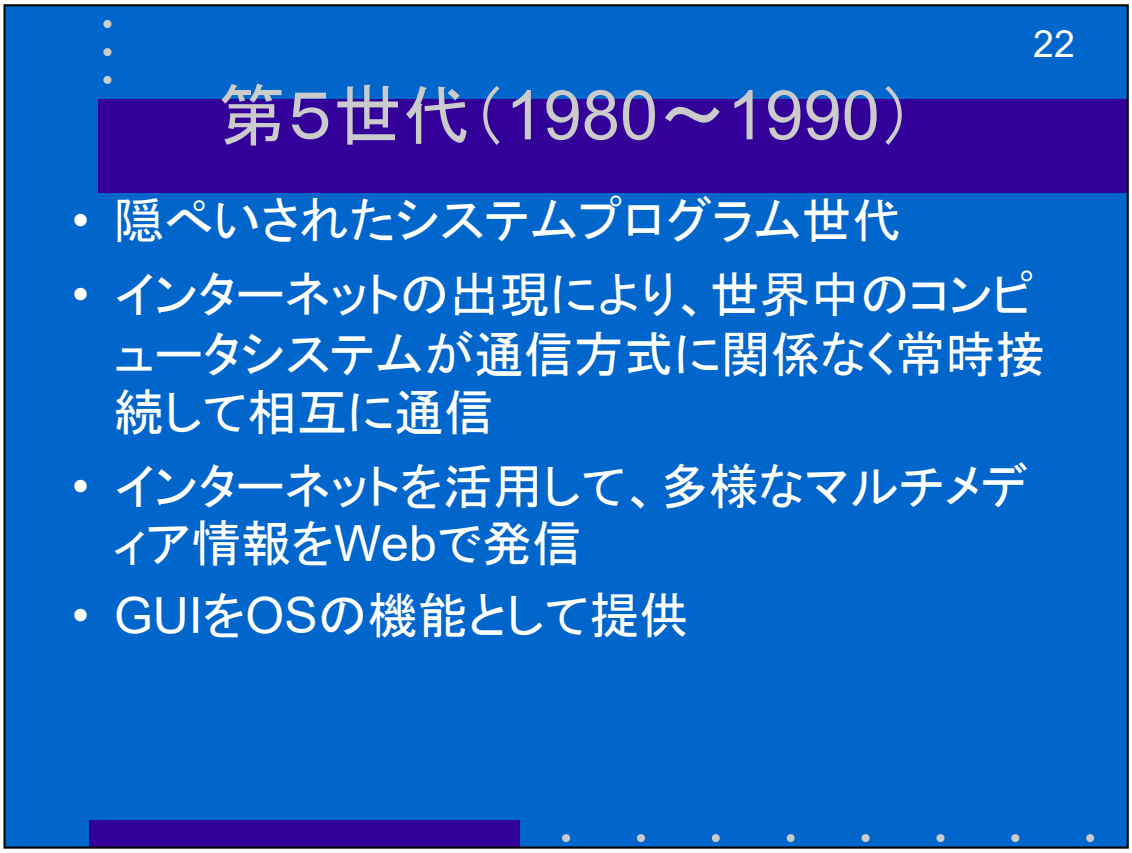



第4世代は、統合プログラミング環境と呼ばれるものが提供されるようになりました。

コンピュータが扱う情報が文字や数値だけでなく、音声や画像などの情報も扱うようになり、情報を伝達する媒体であるメディアが多種多様化したマルチメディアに対応するようになりました。

また、複数のコンピュータシステムをLANで結合して分散処理ができるようになりました。

一方で、このようなメディアの処理方式やLANの通信方式では多数の異なる方式が存在しました。

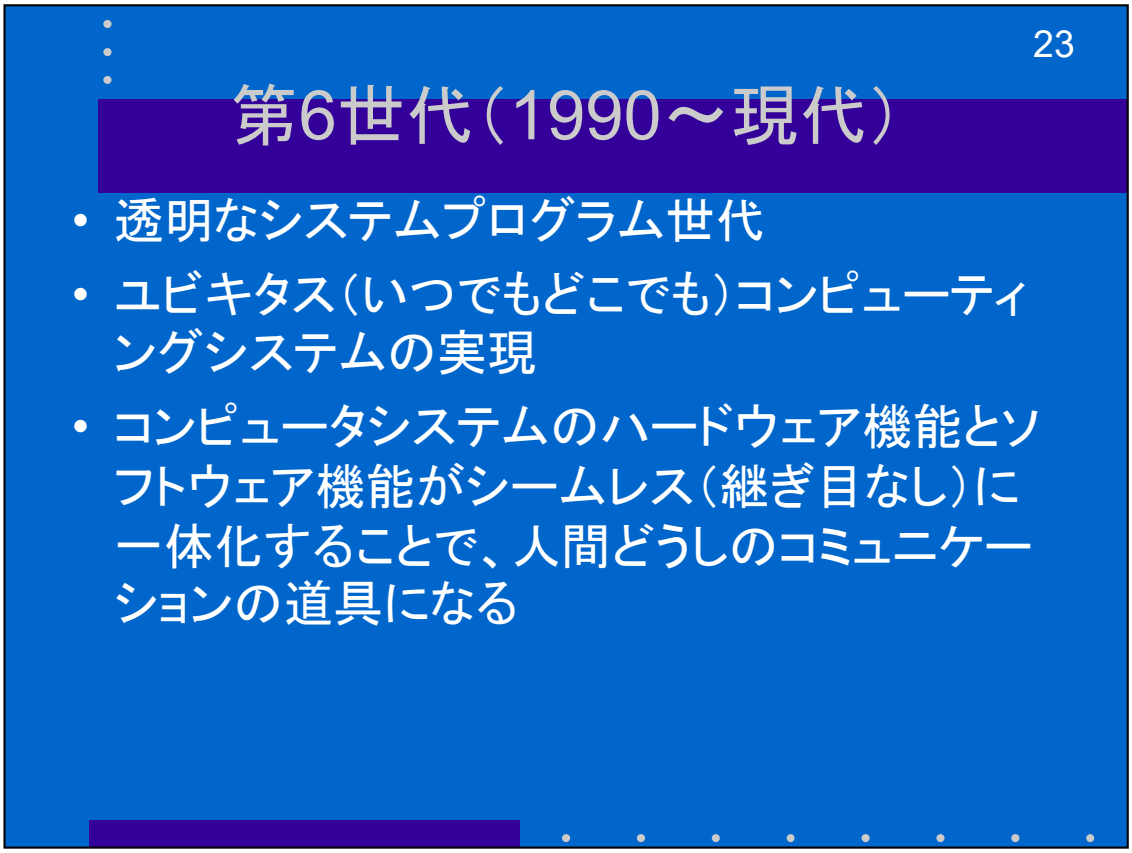



第5世代は、隠蔽されたシステムプログラムの世代です。

ここで「隠ぺい」とは、システムプログラムの持つユーザインタフェース以外の詳細な機能をユーザプログラムからは直接は見えないようにすることを指します。

詳細な機能を見えなくすることで、コンピュータシステムに新たな周辺装置等のハードウェアを加えたときに、システムプログラムの詳細な部分で生じる変更がユーザプログラムからは隠されることになり、ユーザプログラムの修正をしなくてすむようになります。これについては、後日の講義で詳しく説明します。

一方で、インターネットが出現しコンピュータシステムが接続されるようになったり、WebやGUIが登場し、広く使われるようになりました。




第6世代になると、システムプログラムの「隠ぺい」がさらに進み、「透明なシステムプログラム」と呼ばれるようになりました。

第5世代では、システムプログラムが隠ぺいされるようになったと説明しましたが、この隠ぺいはシステムプログラムごとに独自に行われていました。

第6世代になると、システムプログラムの違いを意識しなくてもよいようになり、モバイル端末でも同じ機能が実現できるようになったため、いつでもどこでも同じことができるユビキタスコンピューティングができるようになってきました。

また、透明化により、ハードウェアとソフトウェアの境界が見えなくなって一体化されるようになり、さらにはSNSなど人間どうしのコミュニケーションの道具として使われるようになりました。

# 「OSなし」、「簡易OS」、「本格OS」で何が違うか？

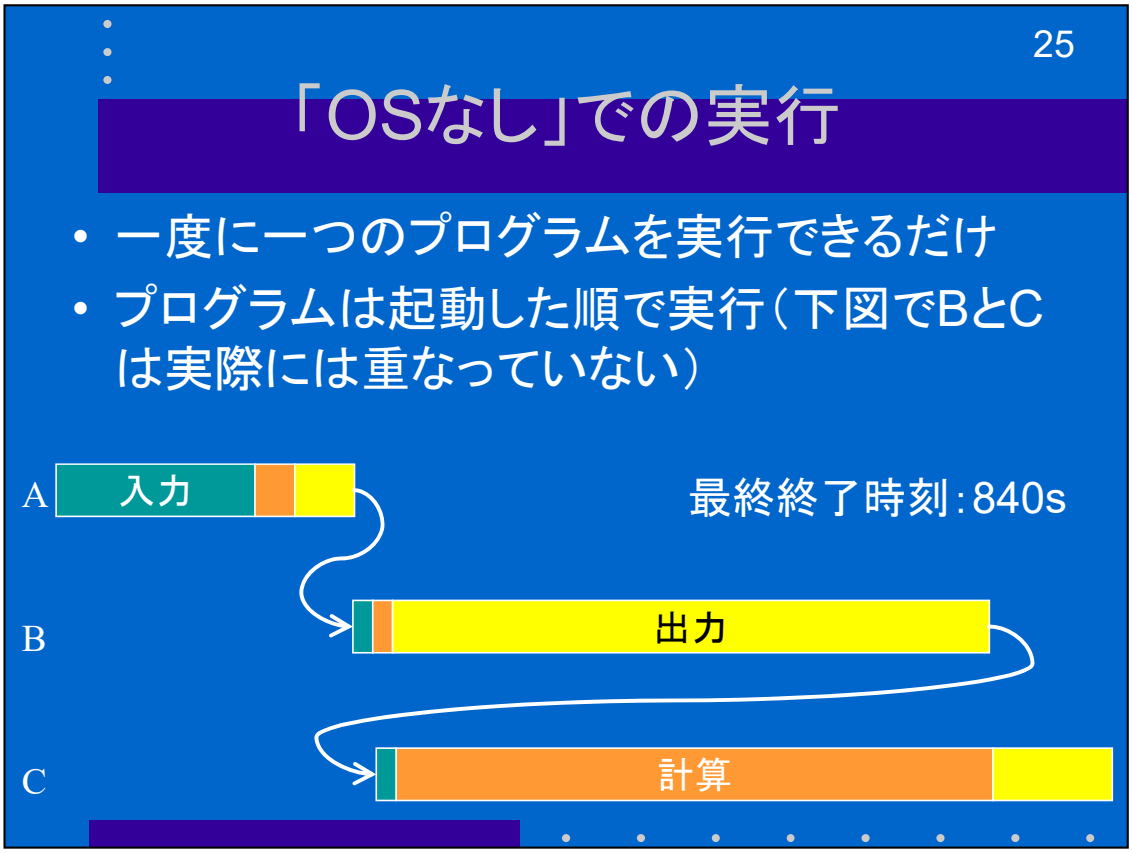


- ユーザプログラムの実行の違いを例に説明する
- 入力時間、計算時間、出力時間の違う3種類のプログラムを考える
  - プログラムA　入力100s, 計算　20s, 出力　30s
  - プログラムB　入力　10s, 計算　10s, 出力300s
  - プログラムC　入力　10s, 計算300s, 出力　60s

それでは、先に説明した、システムプログラムがない、簡易システムプログラム、本格的システムプログラムで、ユーザプログラムの実行がどのように変わるかを具体的に見て行きましょう。

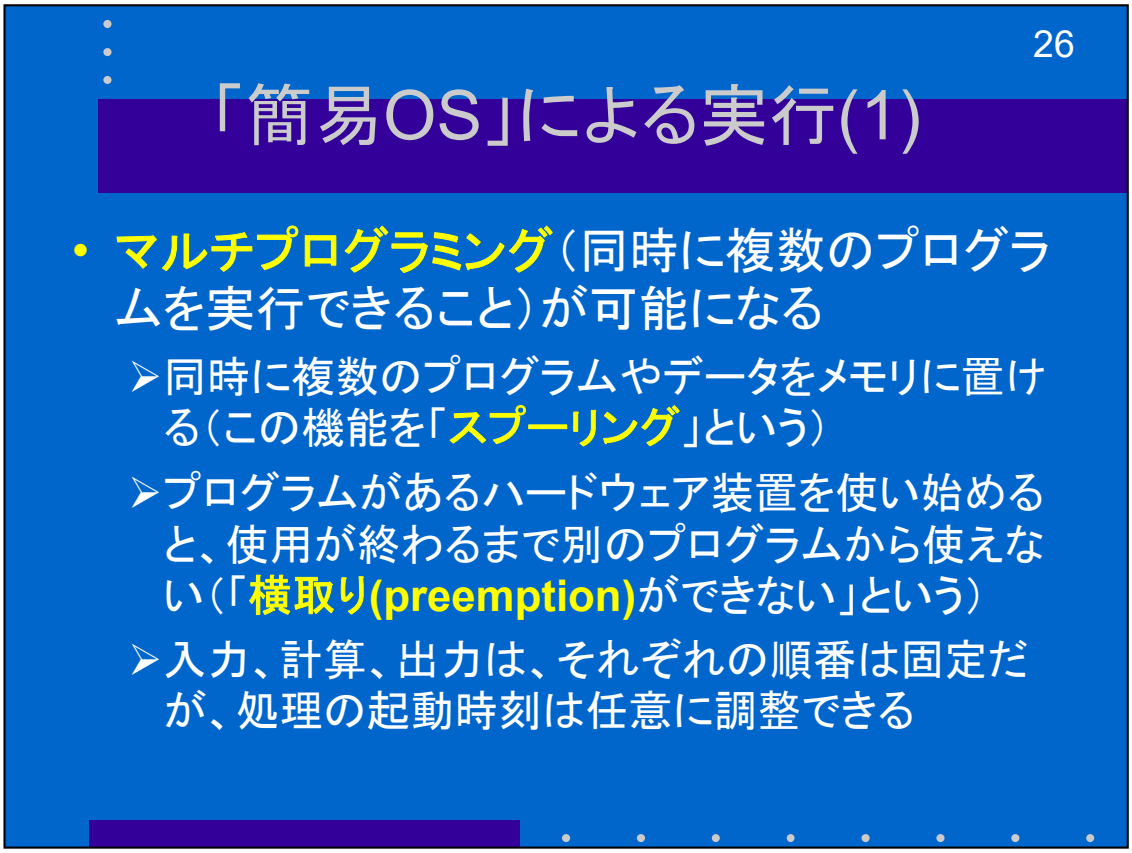
ここでは、簡単のため、システムプログラムとOSを同一視して、単に「OS」と呼ぶことにします。

今、入力、計算、出力のそれぞれの処理ごとにかかる時間が異なる3つのユーザプログラムの実行を考えます。




OSがないコンピュータでは、一つのプログラムの実行が始まると、そのプログラムの実行が終わるまで、別のプログラムを実行することができません。

このため、プログラムは起動した順でのみ実行され、全体の処理時間の総和した時間がかかります。



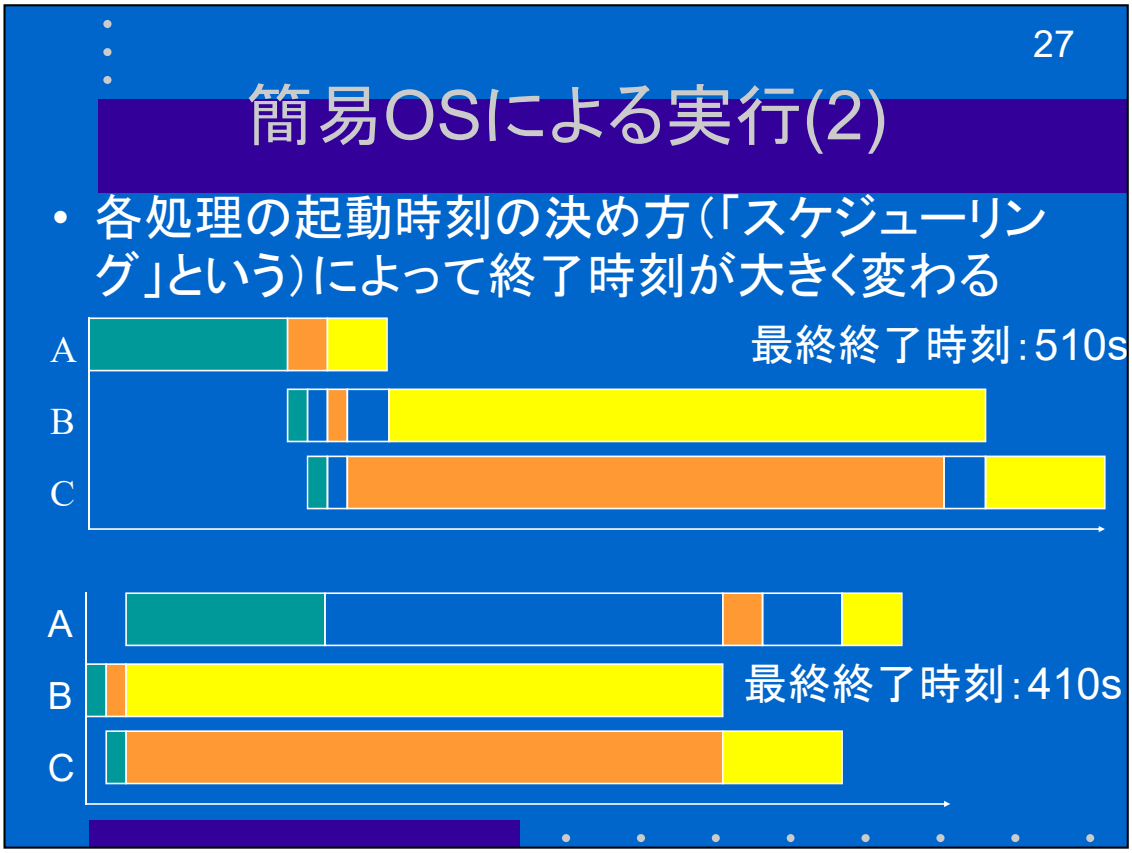

これに対して、簡易OSだと、同時に複数のプログラムを実行するマルチプログラミンができるようになりました。

マルチプログラミンをするためには、まず、複数のプログラムとそれらが使うそれぞれのデータが同時にコンピュータのメモリ上に置けなければならず、この機能をスプーリングと呼びます。

ここで、入力、計算、出力は別々のハードウェア装置で処理されるので、あるプログラムが入力処理をしているときに、同時に別のプログラムが計算処理をすることができるのですが、同じハードウェア装置は同時には一つのプログラムからしか使えず、

しかし、いったんあるプログラムがそのハードウェア装置を使いだすと、そのプログラムがその装置を使う処理が終わるまで、別のプログラムがその装置を使うことはできないものとします。

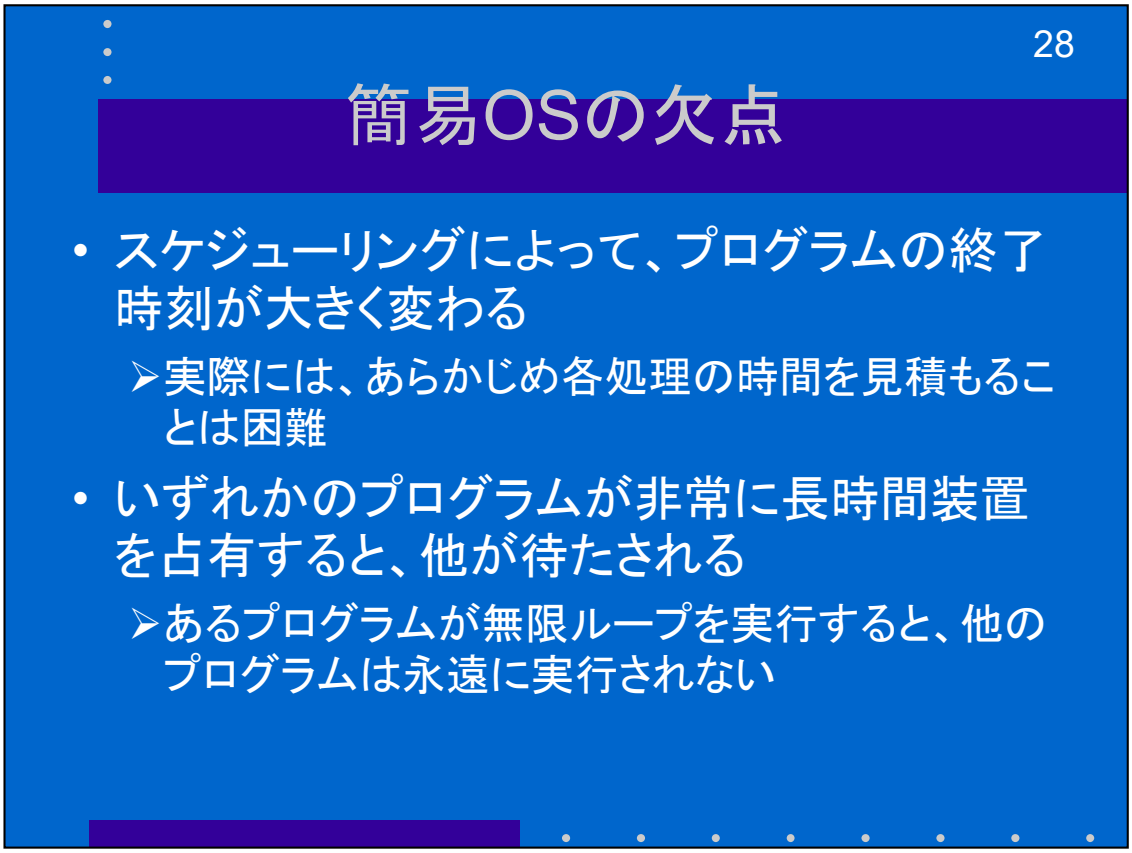
なお、あるプログラムによる装置の使用をいったん中断して、別のプログラムからその装置を使うようにすることを、OSでは「横取り」と呼びます。




簡易OSで、先に述べた3つのプログラムを実行するときは、それぞれのプログラムで装置を使う時刻を調整することで、プログラム全部が終わるのにかかる時間を短縮することができます。

このように、プログラムごとに各処理を行う時刻を決めることをスケジューリングと呼びます。

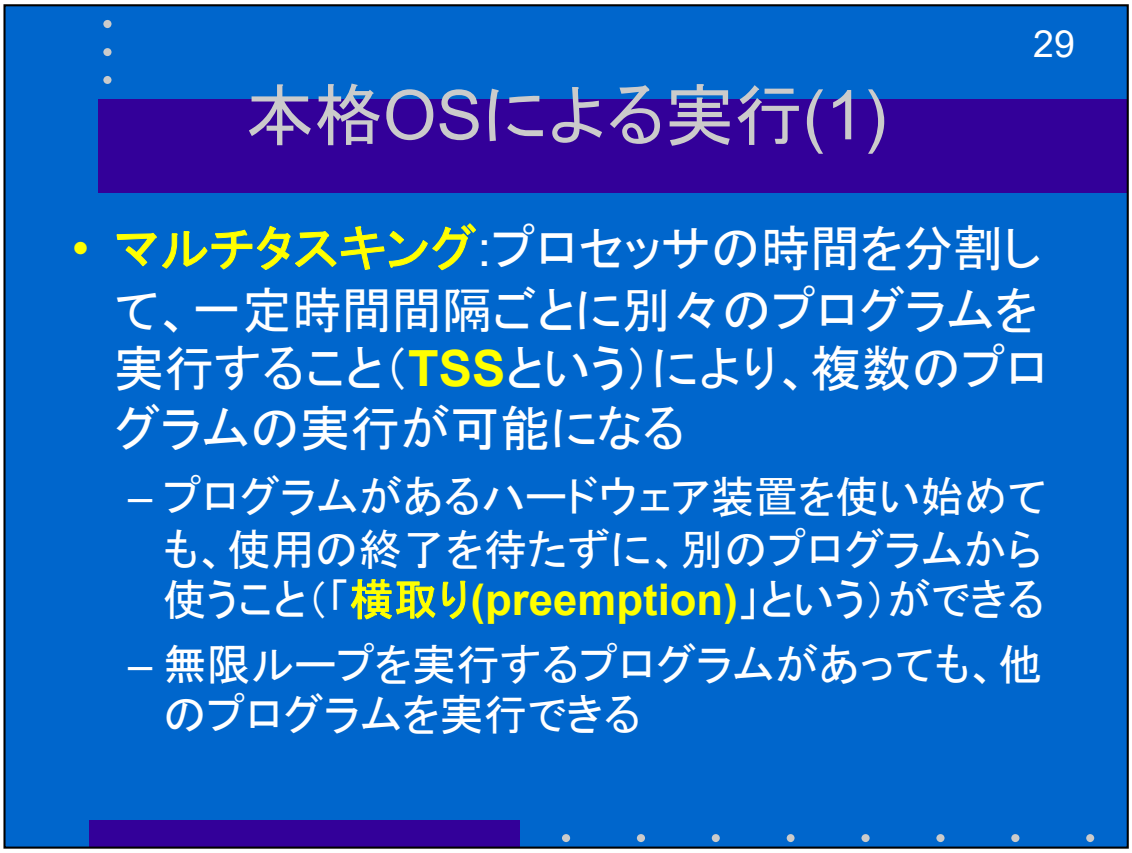
この図からわかるように、スケジューリングの違いで、全体の処理時間が大きく異なります。




簡易OSの欠点は、前のスライドで示したように、プログラムの終了時刻がスケジューリングによって大きく変わることです。

各プログラムで、入力、計算、出力の順番や、それzれの処理にかかる時間を、あらかじめ見積もることができれば、終了時刻を調整することができますが、実際には見積は難しいことが多いです。

さらに、あるプログラムが無限ループを実行してしまうと、他のプログラムは永遠に実行されないといった欠点もあります。




本格OSになると、先に述べた横取りができるようになりました。

このため、プロセッサの時間を分割して、一定時間ごとに別々のプログラムを実行する(これは、先に出てきたTSSのことです)ことで、複数rのプログラムを切り替えながら実行することができることになります。

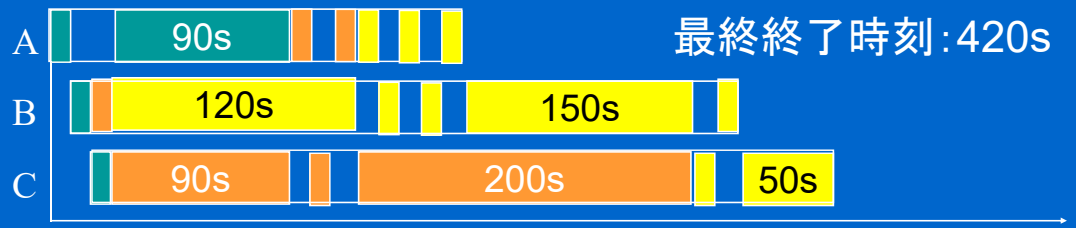
これをマルチタスキングと呼びます。

30

# 本格OSによる実行(2)

以下の条件を仮定する

- すべてのハードウェア装置は横取り可能
- TSSの分割単位（タイムスライスという）は10s（実際のOSではもっと短い）
- 最終終了時刻は、処理の起動順に影響されない




先の3つのプログラムを、本格OSによりマルチタスキングで実行するとこの図のようになります。

このとき、全部のプログラムが終わる時刻は、処理の起動順に大きくは影響されないことになります。

このように、OSなし、簡易OS、本格OSで、同じプログラムであっても、その処理にかかる時間が変わることになります。