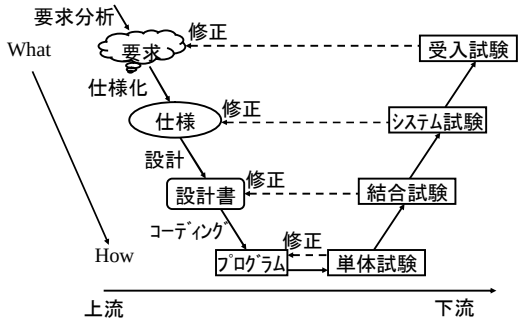
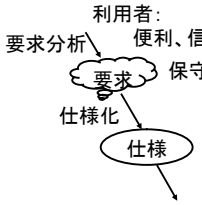
第12回　設計原理とOSの構成法

システム開発



上流工程の困難性

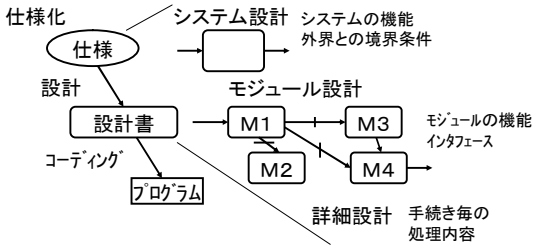


曖昧・矛盾・一般的な解が無い

システムの用途、形態、ハードウェア
利用環境によって大きく異なる

しかし、Whatを誤ると手戻り大

モジュール化

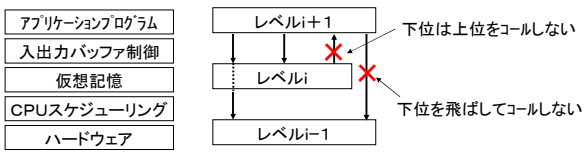


利点
- ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ対象が明確
- ﾓｼﾞｭｰﾙ内なら目が届く
- 変更の影響範囲を限定

考慮要因
- 機能の明確化
- 共通事項の整理
- 独立性

層化接近法（layered approach）

層化接近法
ｼｽﾃﾑの機能を階層に分割、上位層は、すぐ下位層の機能を使用



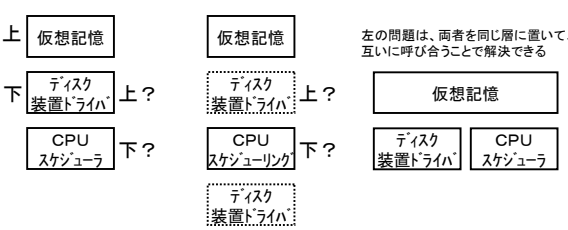
デバッグの容易性（レベルiをデバッグ後、レベルi+1を結合してデバッグ）
- 下位層は、上位層とは無関係にデバッグができる(注)
- 上位層で発生したエラーは、先ず、その層を疑う（下位層は既にデバッグ済）

注：上位層は、要求条件が複雑。仕様が決まり、デバッグできるようになるまでに時間がかかる。上位層を待たずに、下位層のみでデバッグができる。

参考：レベルの定義

①仮想記憶はﾃﾞｨｽｸを使用（ﾄﾞﾗｲﾊﾞをｺｰﾙ）→仮想記憶はﾄﾞﾗｲﾊﾞの上位
②ﾄﾞﾗｲﾊﾞは入出力を開始後、待機中状態に移行するため、CPUｽｹｼﾞｭｰﾗをｺｰﾙ
③PCBが主記憶に入りきれないので、ｽｹｼﾞｭｰﾗはﾃﾞｨｽｸ中のPCBを読み込ませる
②ではﾄﾞﾗｲﾊﾞはｽｹｼﾞｭｰﾗの上位、③ではｽｹｼﾞｭｰﾗはﾄﾞﾗｲﾊﾞの上位

機能間の上下関係は、不明確なものが多い
不必要に階層を分けるとﾊﾟﾗﾒｰﾀ変換等のｵｰﾊﾞﾍｯﾄが増加する



CP/M層構造

MS/DOSの原型

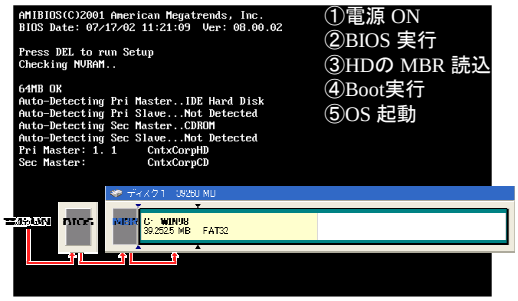
レベル3：AP、ｺﾏﾝﾄﾞ解釈実行ﾌﾟﾛｸﾞﾗﾑ
レベル2：論理（ﾌｧｲﾙ）入出力（BDOS）
レベル1：物理入出力、装置ドライバ（BIOS）
レベル0：ハードウェア

当時のPC用のOSで、
ｼﾝｸﾞﾙユーｻﾞ、ｼﾝｸﾞﾙﾌﾟﾛｾｽのOSとして開発
（多重ﾌﾟﾛｸﾞﾗﾐﾝｸﾞは行わないという前提）

Basic Input / Output System
ｼｽﾃﾑの基本ﾊｰﾄﾞｳｪｱ（CPU、HDD、KB等）を制御するﾏｲｸﾛｺｰﾄﾞ

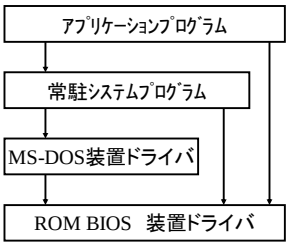
参考　ｼｽﾃﾑ立ち上げ手順

コンピュータの電源を入れてからの基本的な流れ。



MBR：Master Boot Record（ﾊｰﾄﾞﾃﾞｨｽｸの先頭のｾｸﾀ、bootのためのboot）

参考：MS－DOSの層構造

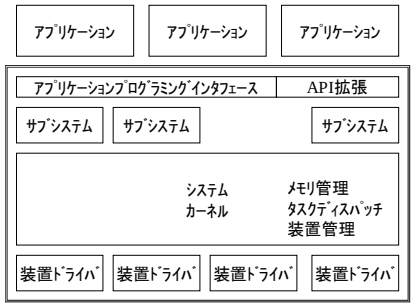


（1）階層化の配慮は無い。
最小の資源でPCに必要な機能を提供するのが目的

例
ｱﾌﾟﾘｹｰｼｮﾝから直接ﾃﾞｨｽｸへ書き込める

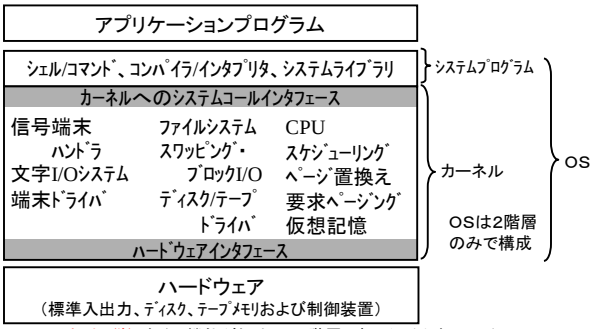
（2）ﾊｰﾄﾞｳｪｱ(8088)の制約
特権ﾓｰﾄﾞ、非特権ﾓｰﾄﾞの区別が無い。
ﾊｰﾄﾞｳｪｱによる保護機能が無い

参考：OS/2の層構造



MS-DOSの後継ｼｽﾃﾑ、ﾏﾙﾁﾀｽｸとﾃﾞｭｱﾙﾓｰﾄﾞ操作を追加
階層構成が明確（利用者が直接下位レベル機能をｱｸｾｽできない）

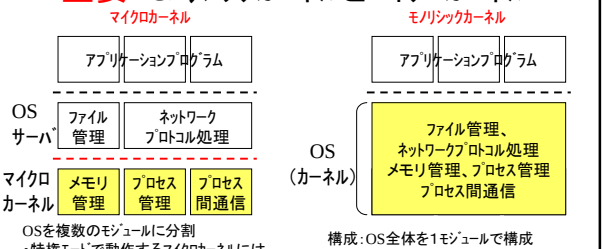
初期のUNIXの層構造



モノリシックカーネル(注)：多くの機能がｶｰﾈﾙの1階層に押し込められている。
（機能追加をしていくなかでｶｰﾈﾙが肥大化）

注：ﾓﾉﾘｼｯｸ＝一枚岩

重要：モノリシックカーネルとマイクロカーネル



OSを複数のﾓｼﾞｭｰﾙに分割
・特権ﾓｰﾄﾞで動作するﾏｲｸﾛｶｰﾈﾙにはﾌﾟﾛｾｽ管理、ﾒﾓﾘ管理、ﾌﾟﾛｾｽ間通信等OSの最低限の機能のみを残す
・他の付加的機能は、非特権ﾓｰﾄﾞで動作するように階層化する
利点：管理が簡単、ﾃﾞﾊﾞｸﾞし易い
欠点：ｵｰﾊﾞﾍｯﾄﾞが大きい

構成：OS全体を1ﾓｼﾞｭｰﾙで構成
利点：効率が良い（ｵｰﾊﾞﾍｯﾄﾞが少ない）
欠点：
・OSの肥大化により、管理が困難（機能の追加・変更が難しい）
・特権ﾓｰﾄﾞ(注)で動作するﾌﾟﾛｸﾞﾗﾑ量が多くﾃﾞﾊﾞｸﾞが難しい（特権命令を使用）

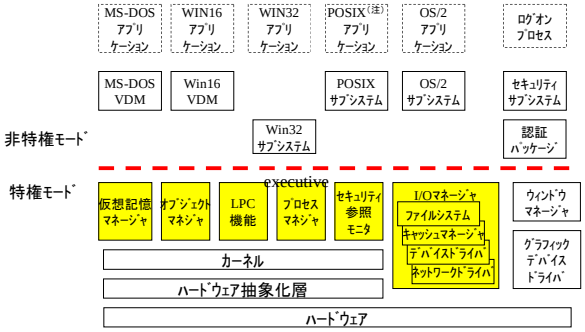
：特権ﾓｰﾄﾞ（ｶｰﾈﾙﾓｰﾄﾞ）　：非特権ﾓｰﾄﾞ（ﾕｰｻﾞﾓｰﾄﾞ）

参考：Windows 95/98/Me

1台のPC上で、MS-DOSのAP、Win16のAP（16ビットマシン用のAP）、Win32のAP（32ビットマシン用のAP）が動くようにした。考え方は仮想計算機だが・・・

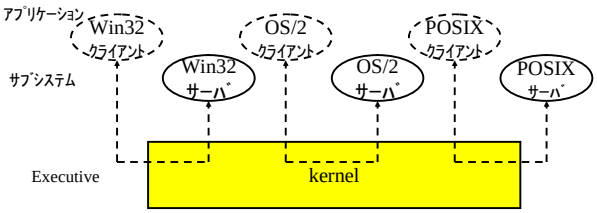
注：カーネルがプリエンプティブでないので、複数のWin16のAPを1プロセスで実行

Windows 7 の構成



注：POSIX（Portable Operating System Interface：Unixの標準の一つ）

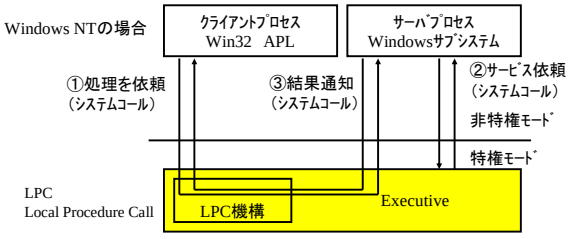
（重要）Windowsのクライアント-サーバ構造



マイクロカーネルでは、アプリケーション（利用者プログラム、クライアント）は、カーネルを経由したメッセージ通信を行うことによりOSのサブシステム（サーバ）の機能を利用する（クライアント、サーバともに利用者モードで動作する）

クライアント，サーバという用語はプロセスの機能分担を指す．コンピュータ間の関係ではない．（普通のPCもサーバになるし，ワークステーションもクライアントになる）

重要：マイクロカーネルのシステムコール



図の例では、クライアントプロセスは、Windowsサブシステム（サーバプロセス）に対して、1回のシステムコールを実行。しかし、実際には、サーバプロセス（非特権モードで動作）もカーネルであるNT Executiveに対し、2回のシステムコールを実行している（計3回(注)）。

注：モノリシックカーネルでは1回のみ。

マイクロカーネルの欠点：オーバヘッドが大きい

（割り込み分析、モード切替、レジスタの退避など種々の処理が必要）。