3 我々一般ユーザはOSとどう関わっ

3–1 ログイン、プログラムの起動、ログアウト

コンピュータ利用の手続き：

① コンピュータ利用の前にアクセス権の確認。

② 次のいずれかの方法でプログラムを起動

- 対話画面におけるコマンド入力
- GUIにおけるアイコン操作。

③ 利用終了をOSに知らせる。

この中でOSがどう関わっているか?

コンピュータ利用の手続き：

① コンピュータ利用の前にアクセス権の確認。OSが確認

② 次のいずれかの方法でプログラムを起動

- 対話画面におけるコマンド入力
- GUIにおけるアイコン操作。

ユーザの会話の相手はシェル/ウィンドウシステム/OS である。

- OSは指定されたプログラムを主記憶に読み込み、そこに制御を渡す。(プログラム実行)
- プログラム実行中は、OSは、ユーザ実行のプログラムが暴走した場合でもシステムをダウンさせることなく運転を継続させる。

③ 利用終了をOSに知らせる。

- ログインからログアウトまでの一連の期間をセッションと呼ぶ。
- 利用者がコンピュータに依頼するひとまとまりの仕事の単位を一般にジョブと呼ぶ。

3–2 コンピュータ起動時・終了時の処理

コンピュータ起動時(電源投入時)の処理

(1) ハードウェアの初期化。

(2) ROMプログラムが起動して、主記憶の初期化、IPL (Initial Program Loader)の読み込みが行われる。

(3) IPLが起動し、OSプログラムを主記憶に読み込む。

ROMプログラムがOSを読み込まない理由:

①ROMを小さくするため。

②ROM処理で不具合発生を抑えるため。

信頼性

(③様々なOSの読み込みを可能にするため。)

使い易さ

(4) OSの初期化。

リセット・再起動時の処理：

- ハードウェアによるリセット

  (1) ハードウェアの初期化。
  (2) ROMプログラムが起動して、主記憶の初期化、IPLの読み込み...
  (3) IPLが起動し、OSプログラムを主記憶に読み込む。
  (4) OSの初期化。

  電源投入の場合と同じ処理だが、処理を開始した時点で電源がOFFからONに変わるのと最初からONになっているのとの違いにより、動作結果が電源投入の場合と違うことがある。
  ⇨ ハードウェアリセットでは不具合から回復しない場合でも、電源投入により不具合から回復することがある。

- ソフトウェアによるリセット

  (3) IPLが起動し、OSプログラムを主記憶に読み込む。
  (4) OSの初期化。

- ソフトウェアによる再起動

  (4) OSの初期化。

正常終了時の処理：

- プロセスの強制終了

デーモンの様に終了を意識して書かれていないプログラムもある。

- メモリ上にあるデータを外部記憶装置に格納する。

プロセスの入出力を高速に行うため、本来なら外部記憶に書き出すべきデータを一旦メモリ上に書き出し、外部記憶へは適当な時期に書き出す様にしているシステムも多い。 効率的な運用

異常終了時の処理：

OSの機能ブロックに障害が発生した場合、障害の内容によって処理内容は様々である。

- OSの異常状態を外部記憶に書き出す。

障害の原因解析を支援するため

- プロセスの強制終了
- メモリ上にあるデータを外部記憶装置に格納する。

信頼性

但し、異常終了時は、OSの動作保証が出来ないため、通常のOS処理とは別にプログラムを用意することが多い。

3–3 装置管理と障害対策

装置管理：

- 接続されている装置の種類と数を管理する。

データ転送をする際の参考になる。 効率的な運用
また、存在していない装置へのアクセスなどの
異常な操作を未然に防げる。 信頼性

- 接続されている各装置の状態を管理する。

障害対策：

接続した装置に障害が発生したら、次のように回復処置が取られる。

(1) 利用中の処理の後処理。

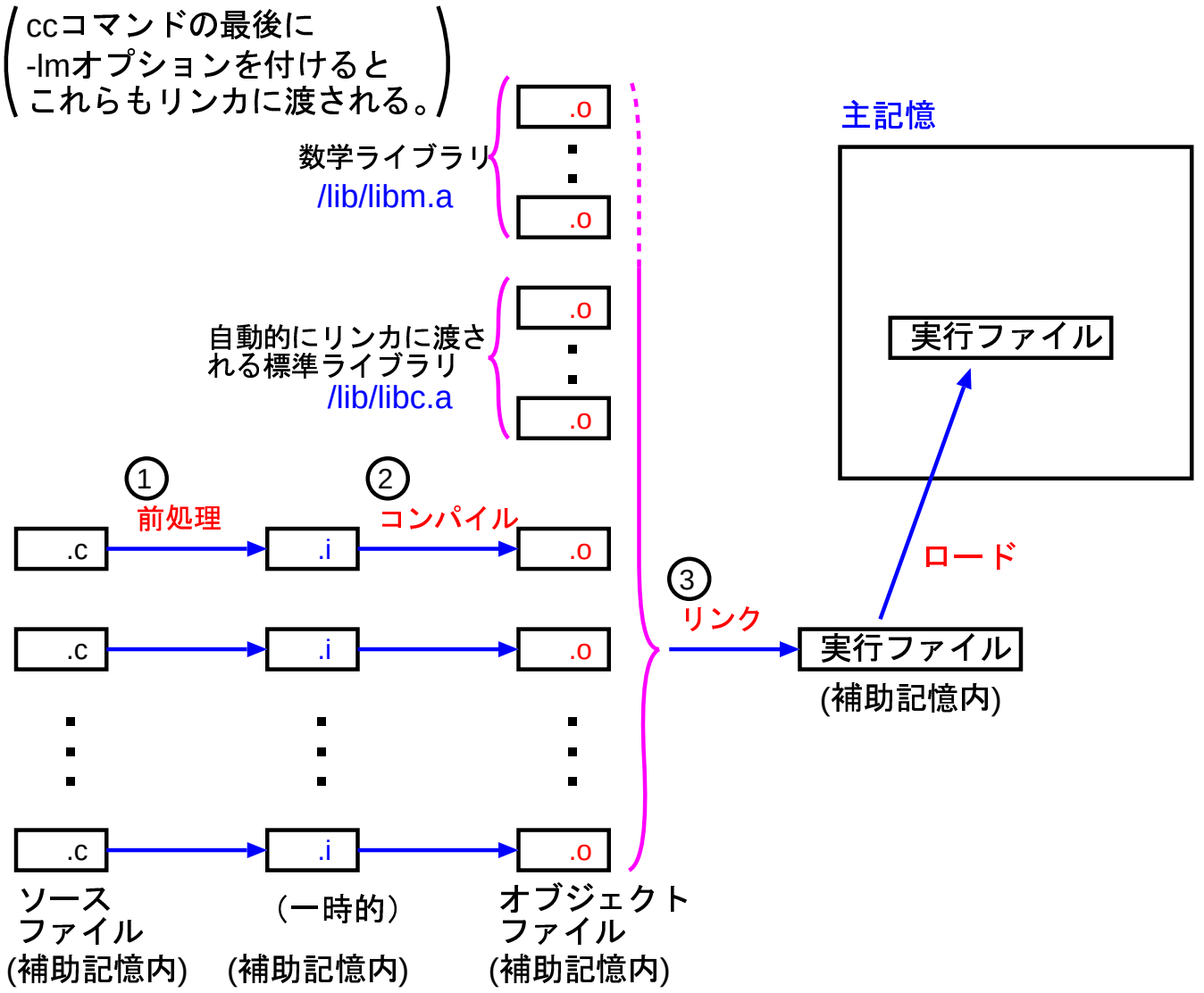
(2) 装置を安定した状態に戻す。 信頼性

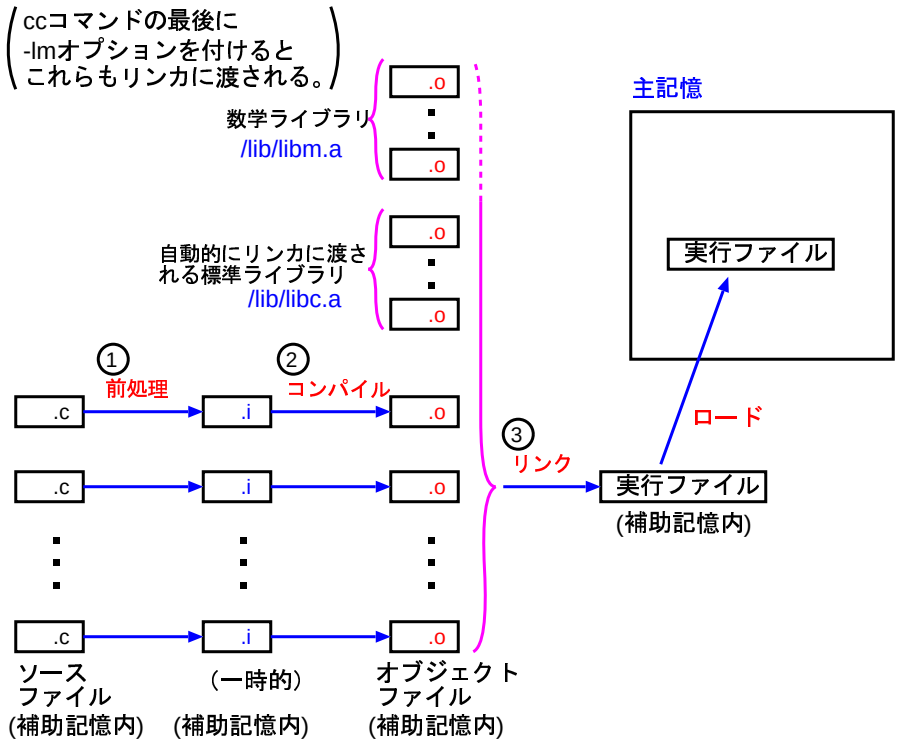
(3) 障害のあった装置を交換する必要がないと判断できる場合は、特定のプロセスからのみアクセスできるようにして試験運転してみる。交換の必要があれば、全面的に装置へのアクセスを禁止した上で交換する。

(4) 装置へのアクセス制限を解除する。

3–4 基礎知識 プログラムの作成・コンパイル・リンク

実行プログラム作成の流れ：





プログラムのリンク：

コンパイルの際extern宣言した変数の部分は然るべき番地に置き換えることは出来ない。

静的リンク ... 通常のgccの様に、コンパイル時にリンクを行う方法。

動的リンク ... ロードセグメントを主記憶に読み込んだ後でリンクを行う。

3–5 基礎知識 実行プログラムの形式

ロードモジュールは次の4 つの部分が並んで出来ている。

(1) ヘッダ部 : 次のような情報から成る。

- 種類識別情報

  テキスト部とデータ部の区別がないロードモジュール、.....
- 作成日付
- テキスト部の先頭番地、大きさ
- データ部の先頭番地、大きさ
- 関係情報部の先頭番地、大きさ

(2) テキスト部 : プログラムの実行コードが並んだ領域である。書き込み不可のメモリ保護を設定する。

(3) データ部 : プログラム内で使われる変数や定数の領域が並んだ領域である。

(4) 関係情報部 : プログラム内で使われる変数の名前と、対応する領域の番地、及び初期値の組み等が保持された領域である。

3–6 基礎知識 プログラムのメモリロード

ロードモジュールプログラムの主記憶への格納・実行は次のように進む。

(1) ヘッダ部を読み込む。

(2) 指定されたものがロードモジュールでないとヘッダ部から判別された場合は、エラーとして処理する。

(3) ヘッダ部を見てロードモジュールの種別を認識する。

(4) プログラム実行に必要なメモリ量
(≈テキスト部の大きさ+ データ部の大きさ+スタック部の大きさ)
をヘッダ部の情報から割り出し、その大きさのメモリ空間を主記憶上に確保する。
スタックは関数呼び出しの際に使う。

(5) ロードモジュールのテキスト部を確保したメモリ空間の中に読込む。

(6) ロードモジュールのデータ部を確保したメモリ空間の中に読込む。

(7) スタックを初期化する。

(8) 実行。

3–7 OSはどんな機能を我々に提供してくれるか

OSの2つの大きな機能：

- コンピュータを使いこなすための様々な操作方法をユーザ／プログラマに提供する機能。
- 資源管理機能。 信頼性, 効率的な運用

3–7–1 OSはコンピュータ利用のためのどんな機能を我々

OSはコンピュータを使い易くするために「裸の計算機」にマクロな機能を付加してプログラマに提供している。
これによって様々な機器の論理化／仮想化／汎用化が実現される。例えば、

- 入出力処理の汎用化 · · · 入出力処理装置を操作するための汎用のインターフェースを与える。
  - ⇨ プログラマは個々の入出力処理装置の特性を知らなくても良い。
  - ⇨ プログラマの生産性が向上。

- ファイルシステムの論理化 · · · ファイル・システムのインターフェースをハードウェアに依存しない論理的なものにする。
  - ⇨ プログラマはファイルを論理的な対象物として扱える。
  - ⇨ プログラマは個々のファイル・システムの特性を知らなくて良い。
  - ⇨ プログラマの生産性が向上。

3–7–2 OSの資源管理機能

コンピュータ資源：

コンピュータにジョブの実行をさせる際は、次の3つの資源を使う。

- 演算装置(CPU) ··· プログラム実行の主体
- 主記憶装置··· 実行中のプログラム等を入れる
- ファイルシステム(外部記憶装置)
  ··· 実行形式プログラム等を長期保存する

実際、 ジョブ／プロセスが実行されるには、

- 主記憶の一部がプロセスに割り当てられ、そこに実行形式のプログラムが格納され、また
- 演算装置(CPU)を使用する権利がプロセスに割り当てられている

必要がある。

資源管理の必要性：

多数のジョブ／プロセスを並行して1つのコンピュータが処理しているので、資源 (特にCPU) の割り当てに関してはジョブ／プロセスの間で競合が起こっている。

⇨ 資源割当ての「交通整理」が必要。 効率的な運用

⇨ OS はコンピュータ資源を適切に管理する使命を負っている。
すなわち、OSは

- 資源の利用状況を常に把握し、
- 場合によっては将来の利用予測も行い、

これらの情報を基に

- 複数のジョブからの資源要求に対して割り付けの順序と時間を決める
  (スケジューリング,scheduling,と言う)

ことによってコンピュータの生産性を最大にしようとする。

各種資源の管理：

次の資源管理方式が有機的に結びついて始めて、システム全体の性能向上・信頼性向上となる。

プロセス管理, または**タスク管理** ...

CPUの管理、すなわち並行して走っているプロセスに対してCPU資源をどういう**スケジュール**で割り当てていくかを考える。 (⇨ 第8節、第12〜16節。)

メモリ管理 ...

主記憶装置の管理、すなわち複数のプロセスにどういう風に主記憶装置を割り当て使っていくかを考える。現在は**仮想記憶**方式が一般的。 (⇨ 第9節、第17節。)

ファイル管理 ...

外部記憶装置の中に**ファイルシステム**をどういう風に構築するかを考える。 (⇨ 第7節、第10節。)