

## 第5章 入出力の制御

花田 英輔

(このPowerPointは渡辺名誉教授作成のものを花田が一部改題した)

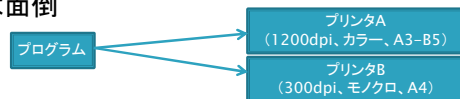
1

### なぜ、入出力をOSで制御するのか？

- ▶ 多数利用者が一つの出力装置を同時に利用すると、**データが混乱**



- ▶ 多様な機器を制御は困難。**様々な例外**を処理は面倒

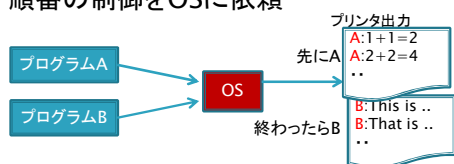


各種トラブル: 紙切れ、インク切れ、電源断、紙詰まり、

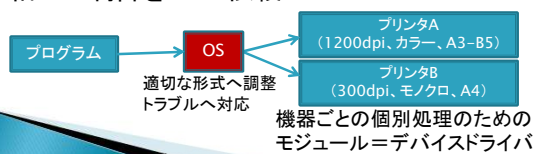
2

### OSで制御すると？

- ▶ 順番の制御をOSに依頼



- ▶ 細かい制御をOSに依頼



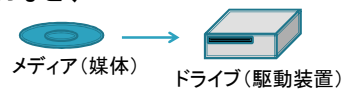
機器ごとの個別処理のためのモジュール=デバイスドライバ

3

### 入出力装置(教科書5.1)

- ▶ 種類

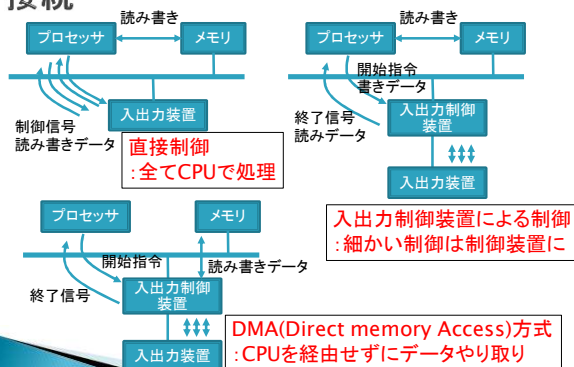
- ハードディスク、CD-ROMドライブ、DVDドライブ、フロッピーディスクドライブ、キーボード、ディスプレイ、マウス、プリンタ、スピーカなど、



- 入出力を略して、I/O (Input/Output)

4

### 接続



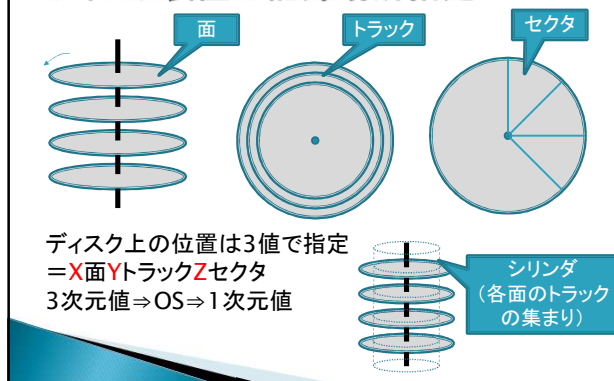
5

### ディスク装置



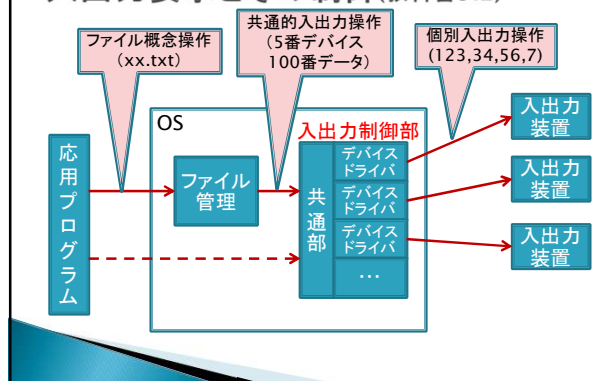
6

## ディスク装置の記録場所指定



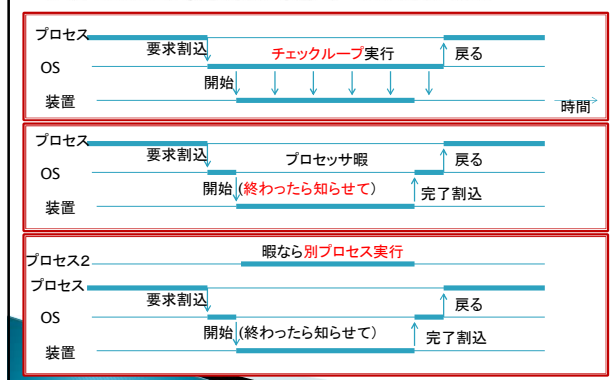
7

## 入出力要求とその制御(教科書5.2)



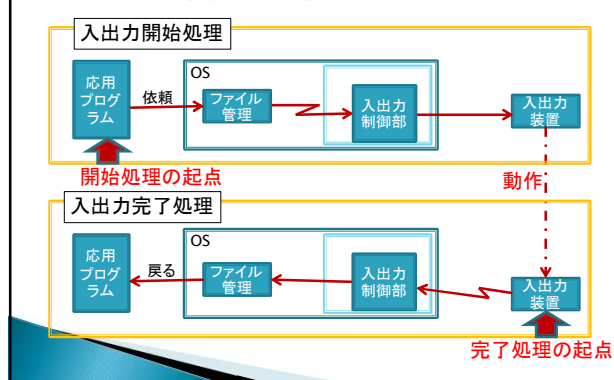
8

## 入出力の開始と完了は別処理



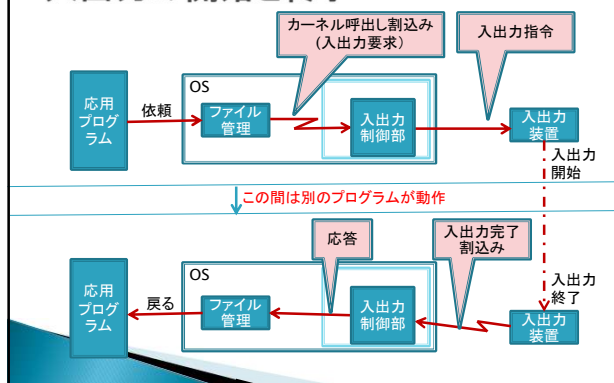
9

## 入出力の開始と完了



10

## 入出力の開始と終了



11

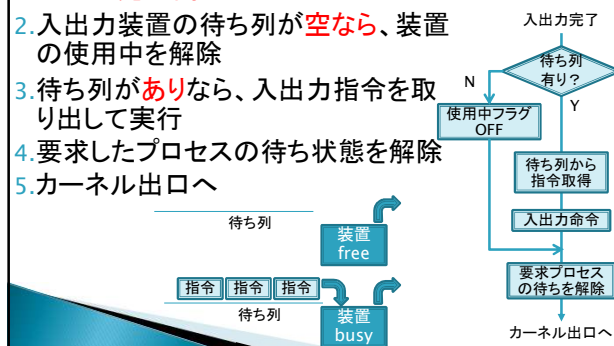
## 入出力起動処理

1. カーネル呼び出し割込み(入出力要求)  
2. 要求したプロセスを待ち状態に  
3. 要求を解析して入出力指令を作成  
4. 入出力装置が空きなら、装置を使用  
中にして指令を発行しカーネル出口へ  
5. 使用中なら、入出力装置の待ち列に  
入れてカーネル出口へ
- 
- ```
graph TD
    Start([入出力要求]) --> WaitProc[要求プロセスを待ちに]
    WaitProc --> ParseReq[要求を解析し指令を作成]
    ParseReq --> Decision{使用中?}
    Decision -- Y --> WaitQueue[指令を待ち列に]
    Decision -- N --> DeviceFree[装置 free]
    DeviceFree --> IssueCmd[指令を発行]
    IssueCmd --> KernelExit[カーネル出口へ]
    WaitQueue --> KernelExit
    DeviceBusy[装置 busy] --> WaitQueue
```
- The flowchart illustrates the process of handling I/O requests. It begins with an 'I/O Request' (入出力要求), leading to 'Waiting for the requesting process' (要求プロセスを待ちに). The next step is 'Parsing the request and creating instructions' (要求を解析し指令を作成). A decision diamond asks 'In use?' (使用中?). If 'Yes' (Y), the instruction is added to the 'Waiting Queue' (指令を待ち列に). If 'No' (N), the device is 'free' (装置 free), and the instruction is 'issued' (指令を発行), leading to the 'kernel exit' (カーネル出口へ). The 'Waiting Queue' also leads to the 'kernel exit'. A 'device busy' (装置 busy) state also leads back to the 'Waiting Queue'.

12

## 入出力完了処理

1. 入出力完了割り込み
2. 入出力装置の待ち列が空なら、装置の使用解除
3. 待ち列が**あり**なら、入出力指令を取り出して実行
4. 要求したプロセスの待ち状態を解除
5. カーネル出口へ

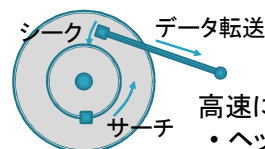


13

## ディスクのアクセス時間

### ディスクのアクセス時間

- = シーク時間 (目的トラックへヘッドを移動)
- + サーチ時間 (目的セクタがヘッド位置に来る回転待ち)
- + データ転送時間 (データを読み書き転送)



高速にするには?

- ヘッド移動を高速に
- 回転を高速に
- 転送を高速に

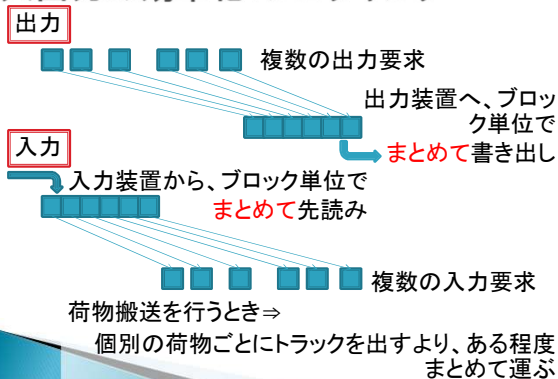
14

## 入出力の効率化(教科書5.3)

- ▶ 入出力の効率化手法は種々ある
  - カーネルの内か外か?
- ▶ 主な手法
  - ブロッキング
  - バッファリング
  - キャッシング
  - ディスク構造自体の工夫

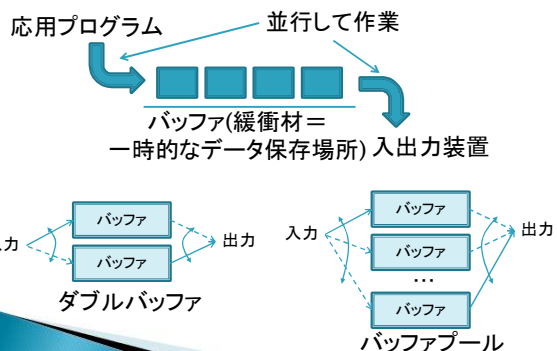
15

## 入出力の効率化:ブロッキング



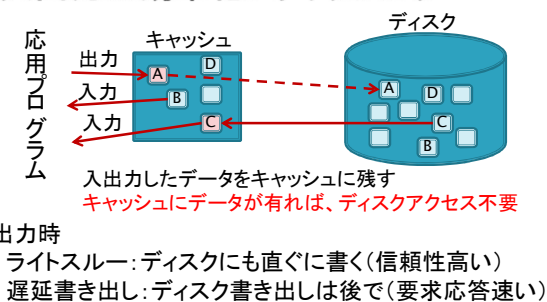
16

## 入出力の効率化:バッファリング



17

## 入出力の効率化:キャッシング



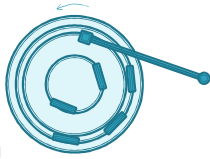
出力時

- ライトスルー: ディスクにも直ぐに書く (信頼性高い)
- 遅延書き出し: ディスク書き出しは後で (要求応答速い)

18

### ディスク入出力の効率化

- ▶ ブロック長を長く
  - 余りに長いと無駄な読み書きも
- ▶ シーク時間を短く
  - 近いトラックの要求から先に処理
  - 一つのファイルは近いトラックに配置
- ▶ サーチ時間を短く
  - 回転に合わせてファイルの順次読み込みできるよう配置



19

### 近年の外部記憶装置(SSDとHDD)

- ▶ 近年はSSDが多く用いられるようになった
  - SSD=Solid State Drive(ソリッド・ステート・ドライブ)
  - HDD=Hard Disk Drive
- ▶ 装置の動作原理
  - HDD:回転する円盤に磁気でデータを読み書き
  - SSD: (USBメモリと同様に) メモリチップにデータを読み書き



20

### SSDとHDDの比較

- ▶ HDDと比較したSSDのメリット
  - 衝撃による故障リスクが低い
  - 読み書きの速度が非常に速い
  - 動作音が静か
  - サイズが小さく軽い
  - デザインの自由度が高い(スティック型等)
- ▶ HDDと比較したSSDのデメリット
  - 最大容量が少ない
    - ・クラウドとの併用で克服可能
  - 大容量になると容量単価が高い

21

### 今回の課題

1. OSでの入出力処理は、開始処理と完了処理の2つに分けられている。それぞれはどこで発生した事象を基にして実行されるか述べよ。
  2. 入出力を開始から完了までを一連のOS処理として実行しない理由を説明せよ。
  3. (予習)「ディレクトリ」とは何か、その内容を含めて調べて記せ
- ▶ 今回のファイル名は“学籍番号-OS06.docx”  
(例: 24238000-OS06.docx)としてください
  - ▶ 締切: 11月14日(金) 18:00 (遅れた場合は減点)
  - 記載時の注意事項**
  - ▶ 参考資料(Webページ)がある場合は**出典を書くこと**
  - **出典を書かずに引用した場合は減点対象です**

22

### 中間試験について

- ▶ 12月1日は**中間試験**を予定しています
  - 出題範囲: **2回めから次回分**まで
    - ・初回からは出題しない予定です
  - **持込みなし**です
  - 毎回の課題と似た問題も出しますが、そうでない問題も有ります
    - ・毎回の課題の正解は、資料を見返してください

23

### 次回のお知らせ

- ▶ 11月17日の講義の後に就職関連のガイダンス「リクナビ活用案内」をします
  - 就職に関する情報を得る手段について説明します
  - 卒業後の進路が進学希望でも、将来就職するのであれば参加すること

24