オペレーティングシステム論
(Theory of Operating Systems)
第二回

ソフトウェア情報学部
基盤ソフトウェア講座
澤本　潤

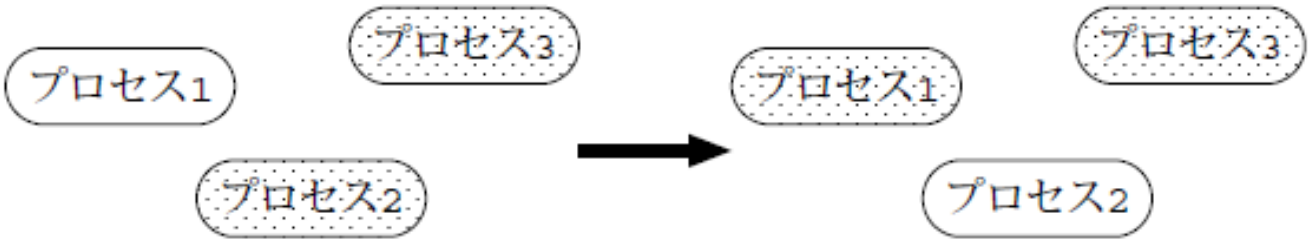
プロセスの概要

プロセス

プロセス(process)

簡単に言えば 「実行 中のプログラム」

タイムシェアリング・システム



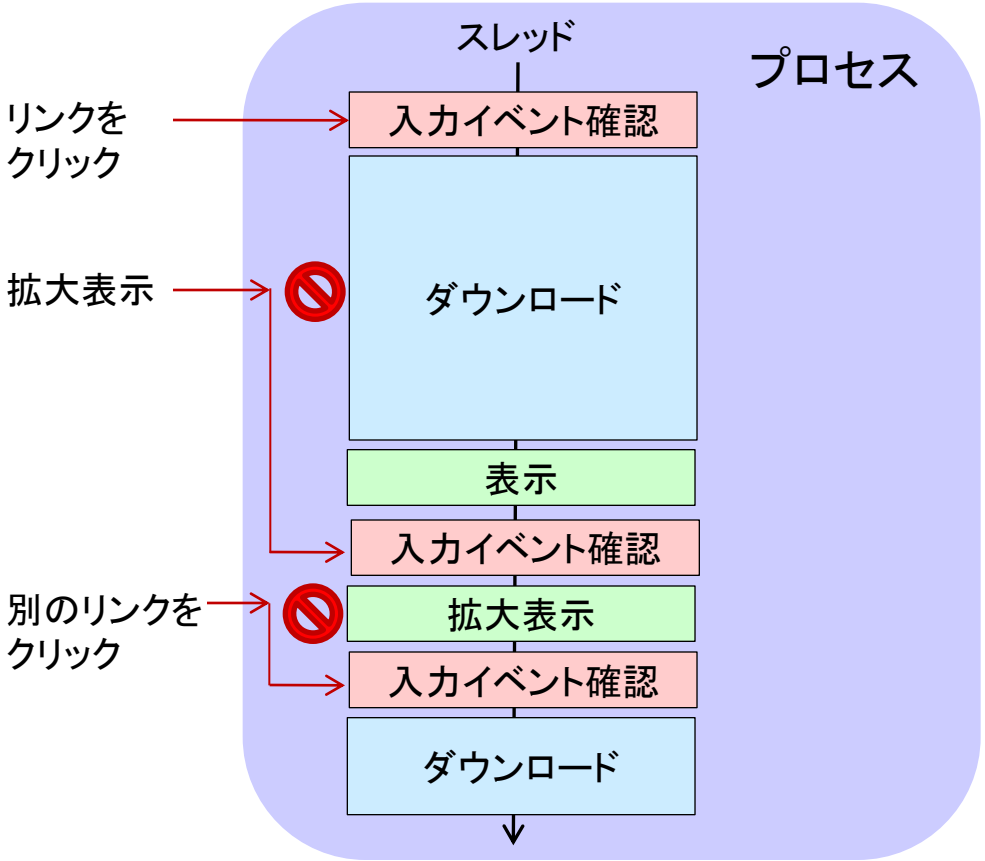
OSによって、中断、再開される。

プロセス管理:
(1)プロセス割り付け:プロセスをメインメモリに割り付ける(空間的)
(2)プロセッサ割り付け:プロセッサの実行時間をプロセスに割り付ける(時間的)

スレッド（軽量プロセス）

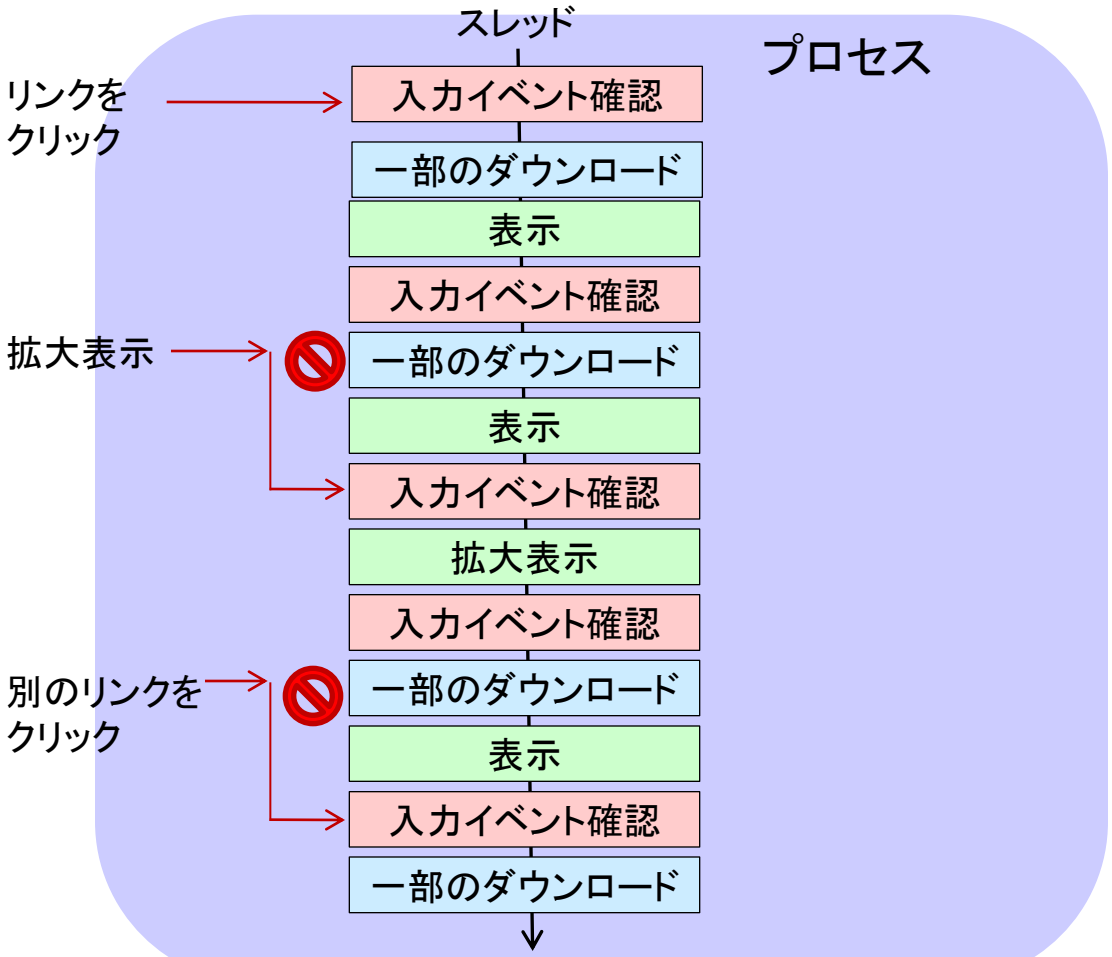
- プログラムの処理の流れのこと。

例 1 単純な Web ブラウザの実装 (シングル スレッド)



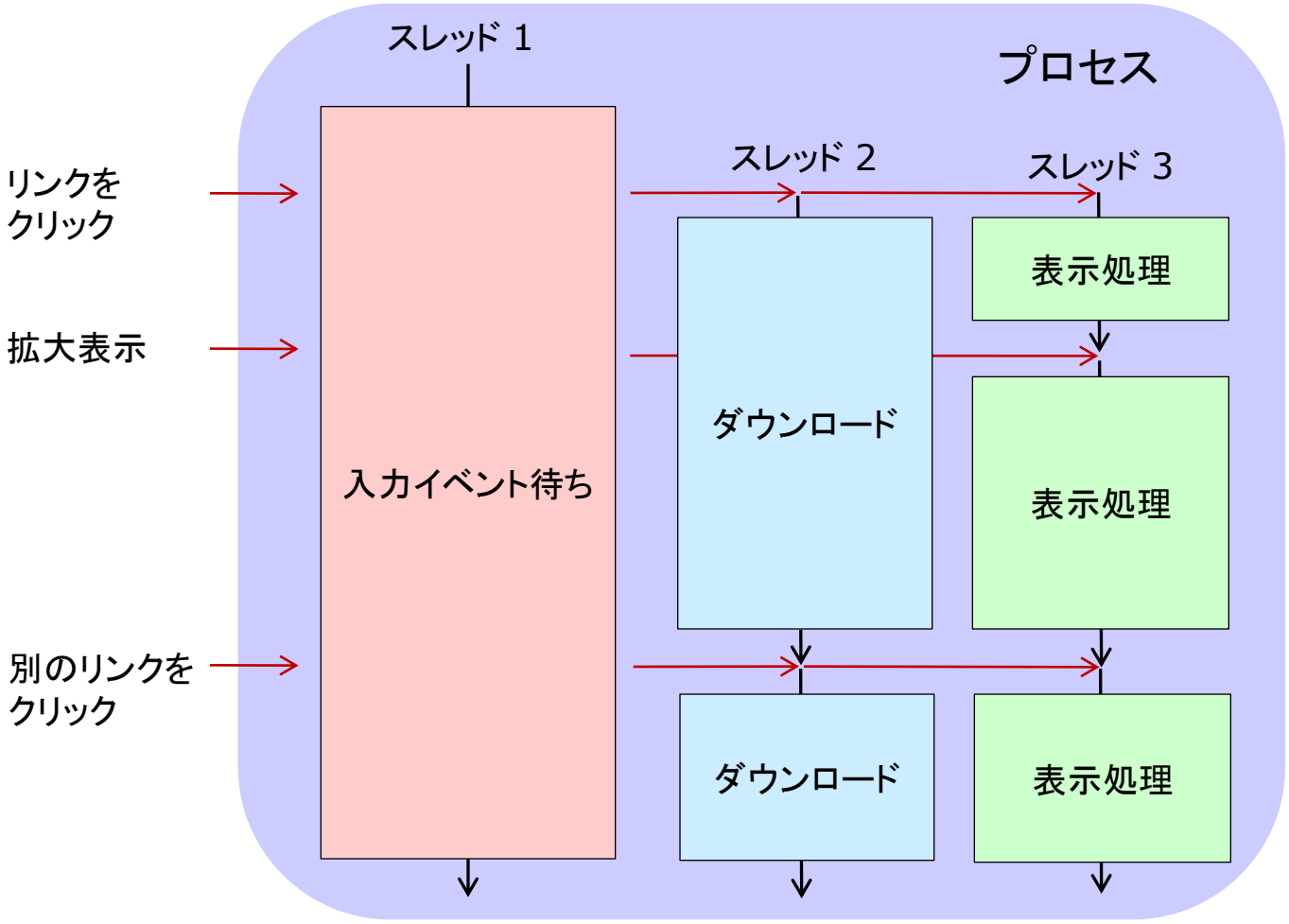
スレッド

例 2 シングル スレッドによる工夫した実装



スレッド

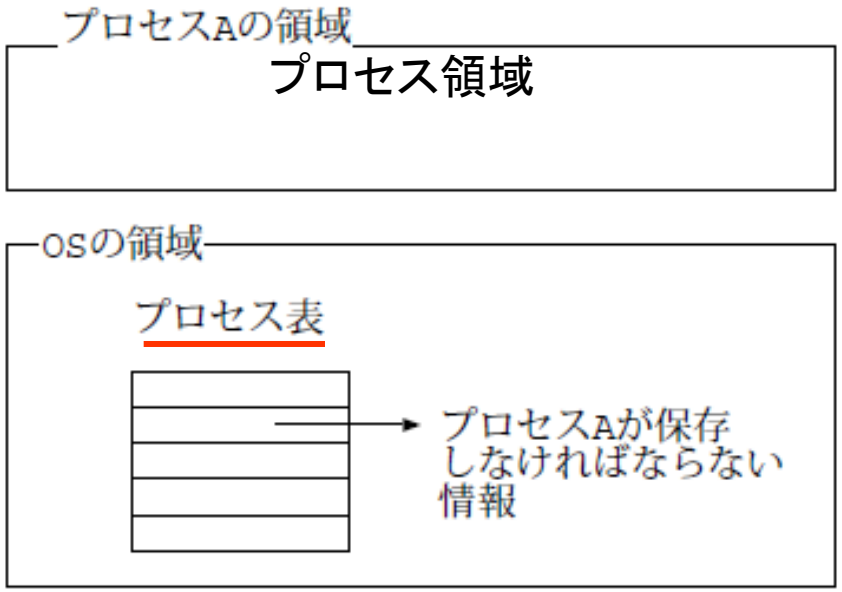
例 3 マルチスレッドによる実装



プロセス領域と PCB（プロセス制御ブロック）

プロセス表

プロセスの中断、再開を実現するためには、
プロセスを中断する時に、再開できるための情報を
保存する必要がある。

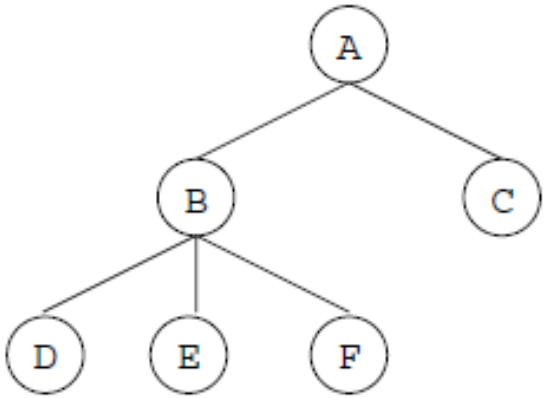


プロセス表　＝　PCB（プロセス制御ブロック）

プロセスの生成

プロセスは、さらにプロセスを生成できる（→ 子プロセス）

親子関係のプロセスが木構造状に表現される。



割り込みとは

- ユーザプログラムおよびハードウェア装置のそれぞれからOSへの多種多様なOS業務の依頼、あるいは、そのためのOSとの通信を、統一的に行う仕組み
- プログラムの実行を、予想できない事態や不測の事態の発生時に、強制的かつ動的に変更する仕組み

割り込み (interrupt)

「割り込み」が生じると、

現在実行している処理から、別の処理へ分岐する。

ハードウェアが提供する機構である。

割り込みの要因を「例外」(exception) ということもある。

※ 手続き呼び出しは、プログラムカウンタの書き換えを利用しているだけで、CPUにとっては連続した通常の実行に過ぎない。
でもTRAPは、「割り込み」の一種なので、ちょっと違う!

割り込みの必要性

- 不測の事態：プログラムに記述できない
- 異常や例外の検知：異常、エラー、例外
- ハードウェア装置→OS　通信
- ユーザプログラム→OS　通信
- 競合するハードウェア利用要求の調停：プロセッサ時間スケジューリング、メモリ領域管理
- 非同期動作しているハードウェア間の通信：入出力割り込み

割り込み要因

- 内部割り込み　…　内部装置でのプログラム実行の結果発生する割り込み ― ソフトウェア割り込み
  - 暗黙的
    - 命令実行例外
      - メモリアクセス例外、特権命令違反、メモリ保護違反、不正命令、不正オペランド、演算例外（ゼロ除算、オーバフロー）
  - 明示的
    - SVC：システムコールによる特権モード移行
    - ブレークポイント（トラップ命令）
- 外部割り込み　…　外部装置でのプログラム実行とは非同期に発生する割り込み ― ハードウェア割り込み
  - ハードウェア障害
  - リセット
  - タイマ割り込み
  - 入出力割り込み

多重レベル割り込み

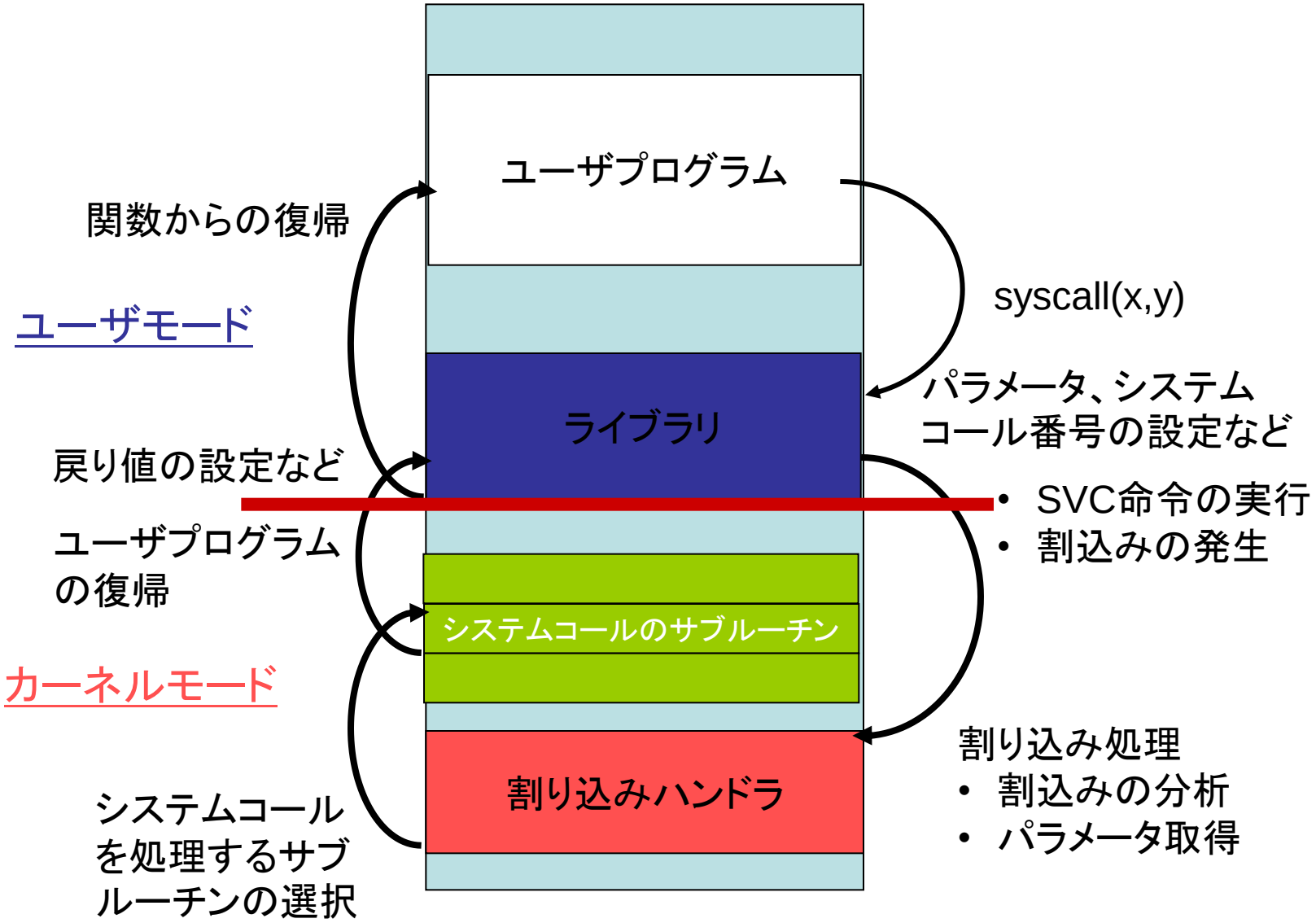
- ある割り込み処理中に、別の要因の割り込みが発生
- 割り込み処理の優先度（緊急性）
  - 処理中の要因よりも高優先度割り込みは優先的に受け付ける
  - ハードウェア障害→リセット→命令実行例外→入出力割り込み→SVC→ブレークポイント命令
  - メモリアクセス例外、メモリ保護違反→演算例外
  - 高速入出力装置からの割り込み→低速入出力装置からの割り込み

割り込み処理

0. 割り込みの発生
1. 割り込みの受付
2. 割り込み禁止状態への移行
3. ハードウェア状態の退避（PSWの退避）
4. 割り込み要因の識別
5. 割り込みハンドラへの分岐
6. ソフトウェア状態の退避
7. 割り込みハンドラによる要因ごとの処理（狭義の割り込み処理）
8. ソフトウェア状態の回復
9. ハードウェア状態の回復
10. 割り込み可能状態への移行
11. 割り込み受け付け時点への復帰

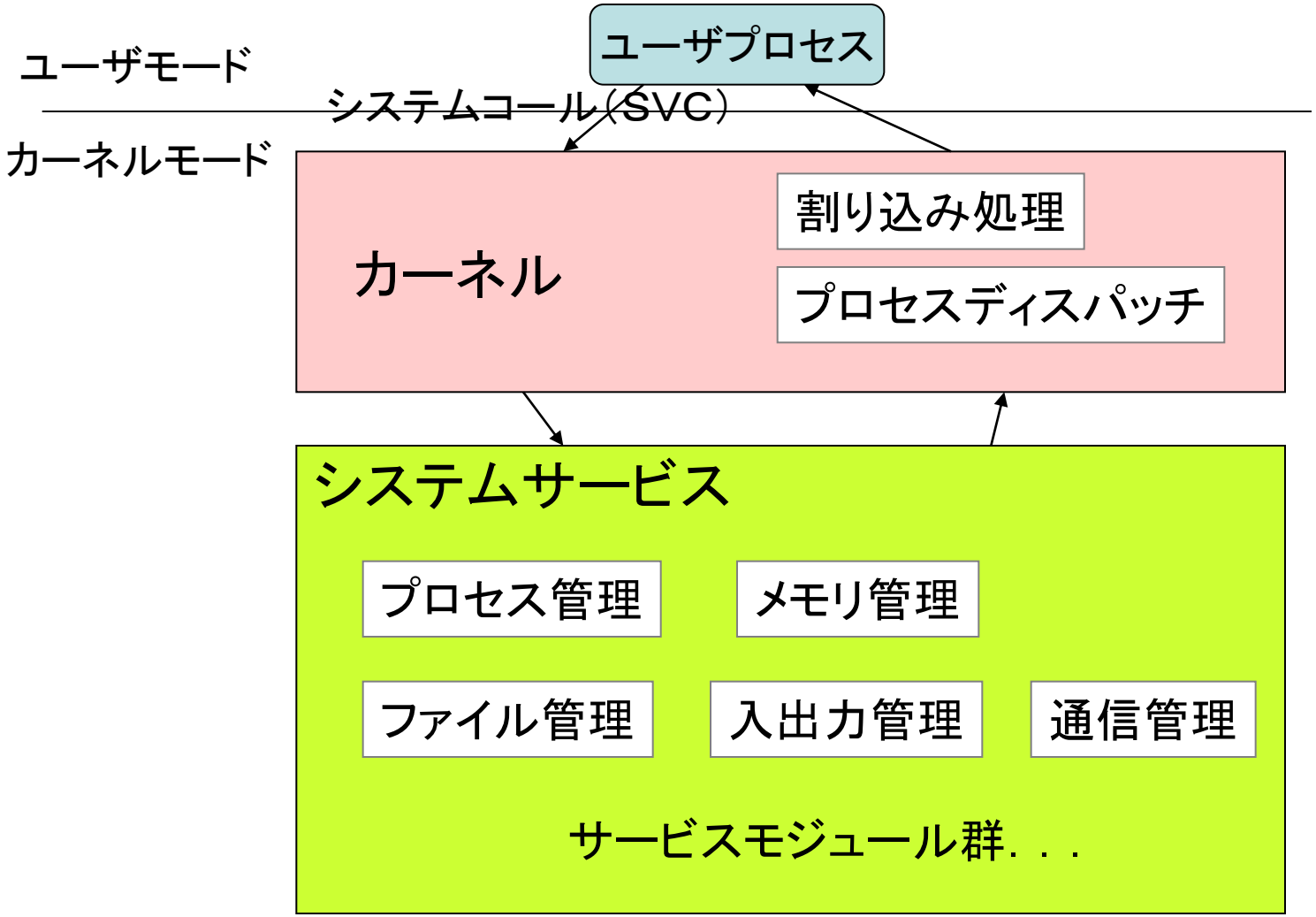
6～8：割り込みハンドラ

システムコールを例とした割込み



www.am.ics.keio.ac.jp/members/morimura/teu_os/OS_3.ppt

OSのモジュール構成（概略）



オペレーティングシステムとのインタフェース

ユーザプログラム

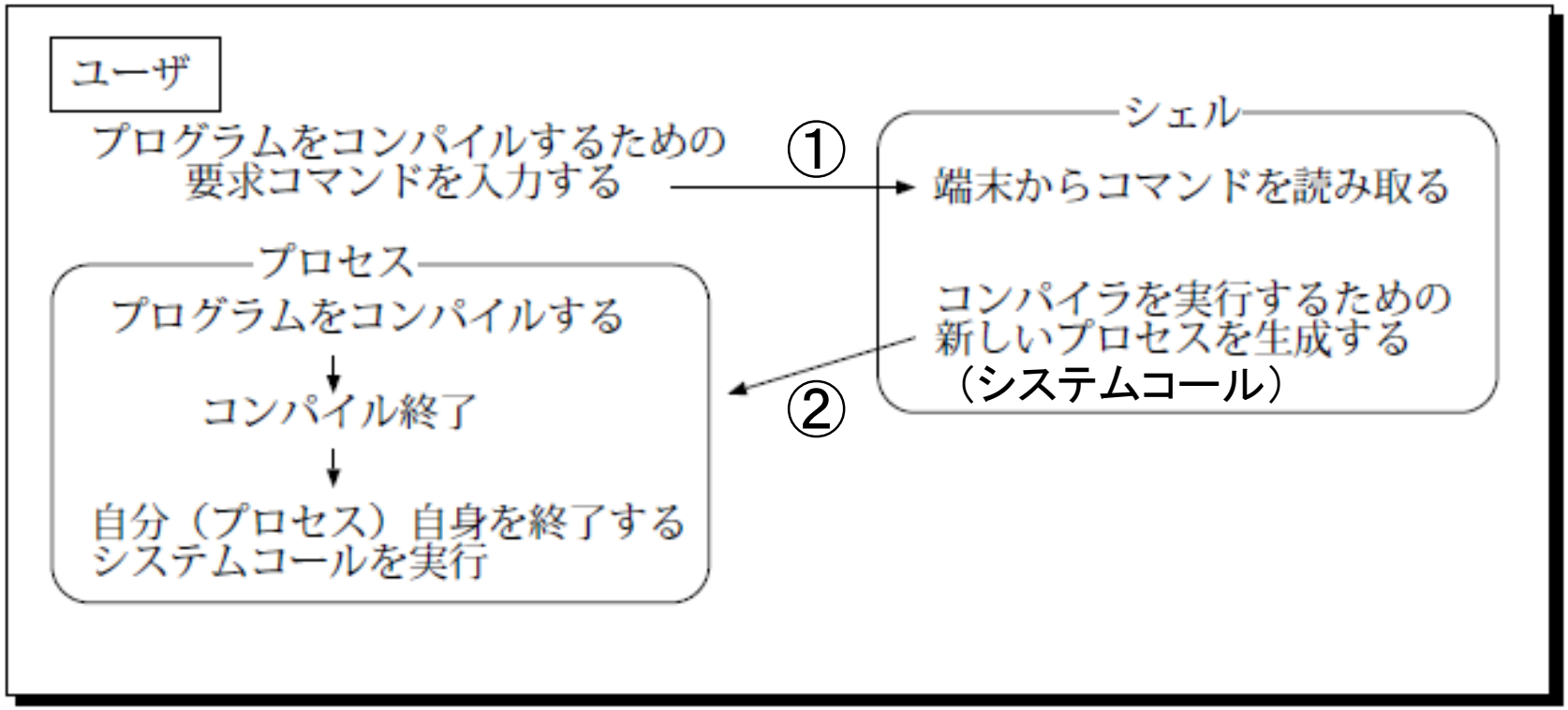
↓

OS

システムコール (system call)
(OSが提供する拡張命令の集合)

UNIXシステムコールによって、
生成、削除、利用される対象は
... プロセス 、ファイル

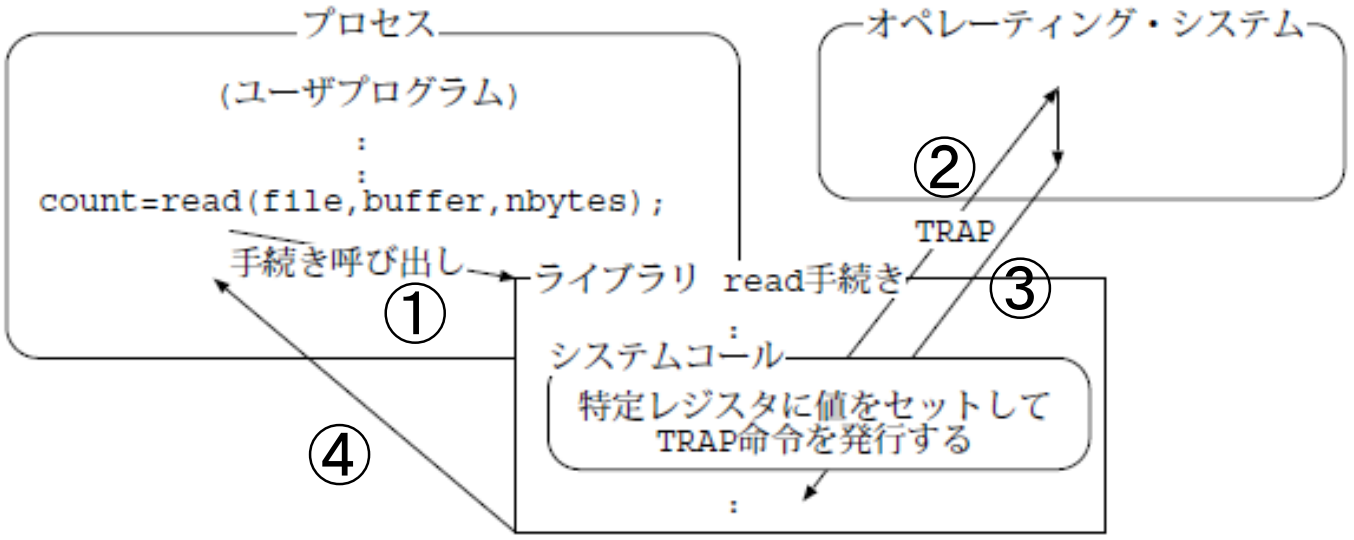
コマンド・インタプリタ－システムコールの例



システム・コール (system call)

システム・コールを発行することによって、オペレーティング・システムへ通信し、サービスを要求する。

- システム・コールに該当する手続きが、ライブラリ内に（ユーザプログラムから呼び出せる形式で）用意されている。



シェル上でのキーボード割り込み

こんなこと知ってますか？

シェルからプログラム実行中に、キーボードから

C-C(コントロールを押しながら'C') で、割り込みを発生させてプログラムを中止（途中で終了）させることができる。

C-Z(コントロールを押しながら'Z') で、同様に中断できる。この場合は、後で再開することができる。

```
ISUXC033[97]% sleep 100
^C                              ← C-Cで中止した
ISUXC033[99]% sleep 100
^Z                              ← C-Zで中断した
ISUXC033[100]% fg               ← 再開した
sleep 100
```

シェル

シェルは、OSの一部ではないが、OSの機能を頻繁に使う。

- ユーザがログインすると、シェルが起動される。
- シェルは、端末を標準入力と標準出力として割り当てる。
- "$" や "%" をプロンプトとして出力し、シェルが「コマンド入力待ち」であることをユーザに示す。
- ユーザが "date" と入力すると、シェルが子プロセスを生成し、そこでdateプログラムが実行される。 子プロセス実行中は、シェルはその実行が終了するのを待つ。
- 子プロセスが終了すると、シェルはプロンプトを再び出力して、次の入力を待つ。

注）コマンドによっては、子プロセスを作らずに自プロセス上で実行するものもある。

コマンドの最後に&をつけると、シェルはコマンドの実行終了を待たないで、直ちにプロンプトを表示する。（バックグラウンド・ジョブ）

ユーザ・モードとカーネル・モード

カーネル・モード：

すべての命令を使うことができる、

何でもアリのモード。

（スーパーバイザ・モードとも呼ばれる。）

ユーザ・モード：

入出力や特別な命令が使えない、制限されたモード。

プロセス管理

（1）プロセス割り付け：プロセスをメインメモリに割り付ける（空間的）

（2）プロセッサ割り付け：プロセッサの実行時間をプロセスに割り付ける（時間的）

オペレーティング・システムは、
どのように設計され、構成されているか。

↓

オペレーティングシステムの中核をなす概念

↓

プロセス
（実行中のプログラムを抽象化したもの）

概念としては（プログラムは料理のレシピで、
プロセスは実際に料理をするという行為。）

多重プログラミング・システム を思い出そう。

↓

複数のプログラムが、
あたかも並列に動作しているような錯覚
疑似並列 (pseudoparallelism)

↑

CPUがプログラムからプログラムへ、
数十ミリ秒間隔で切替え
（1ミリ秒　＝　1／1000秒）

→ このような並列な動作を容易に扱うために、 ....

マルチタスキング　＝　マルチプログラミング（多重プログラミング）
≠　マルチプロセッシング

プロセッサとプロセス（タスク）との対応を、「1プロセッサ対多プロセス」で多重化し、時分割制御によってプロセス（タスク）を切り替えながら、プロセッサをはじめとする各種ハードウェア機構を共用するOSのプロセッサ管理・制御方式

プロセス・モデル

オペレーティングシステムを含む
すべてのソフトウェアは、
いくつかの（逐次）プロセスで構成される – とする。

↑

実行しているプログラム自身。

プログラムカウンタ、レジスタ、そのときの変数の値を含む。

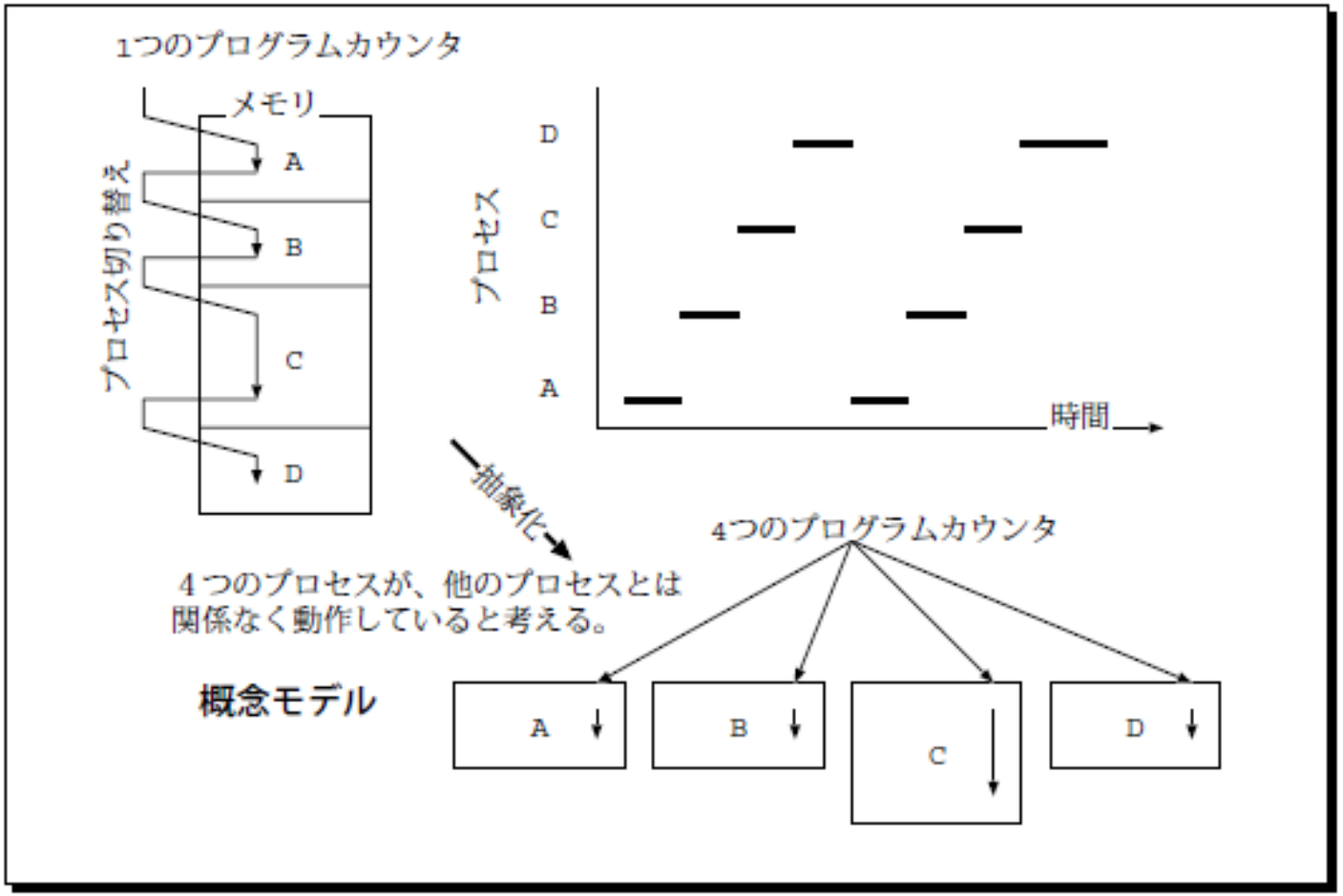
（「プロセス = 仮想のCPU」 ... と考える）

↑

現実には、一つのCPUが プロセスからプロセスへ切替えている

多重プログラミング (multiprogramming)

多重プログラミング


1つのプログラムカウンタ
メモリ
A
B
C
D
プロセス切り替え
プロセス
D
C
B
A
時間
抽象化
4つのプログラムカウンタ
4つのプロセスが、他のプロセスとは
関係なく動作していると考える。
概念モデル
A
B
C
D

UNIXにおけるプロセス生成

FORKシステムコール：

プロセス生成のためにUNIXで提供されている。

呼び出し側のプロセスと全く同一のコピーを作成する。

```
forktest()
{
  int pid;
  printf("Start of test\n");
  pid = fork();
  printf("Returned %d\n",pid);
}
```

プロセスの2種類の停止状態

- プロセスが論理的に実行できずに停止 **（ブロック状態）**

  例）プロセスが他のプロセスと相互作用するとき

  cat *chapter*1 *chapter*2 *chapter*3 | grep *tree*

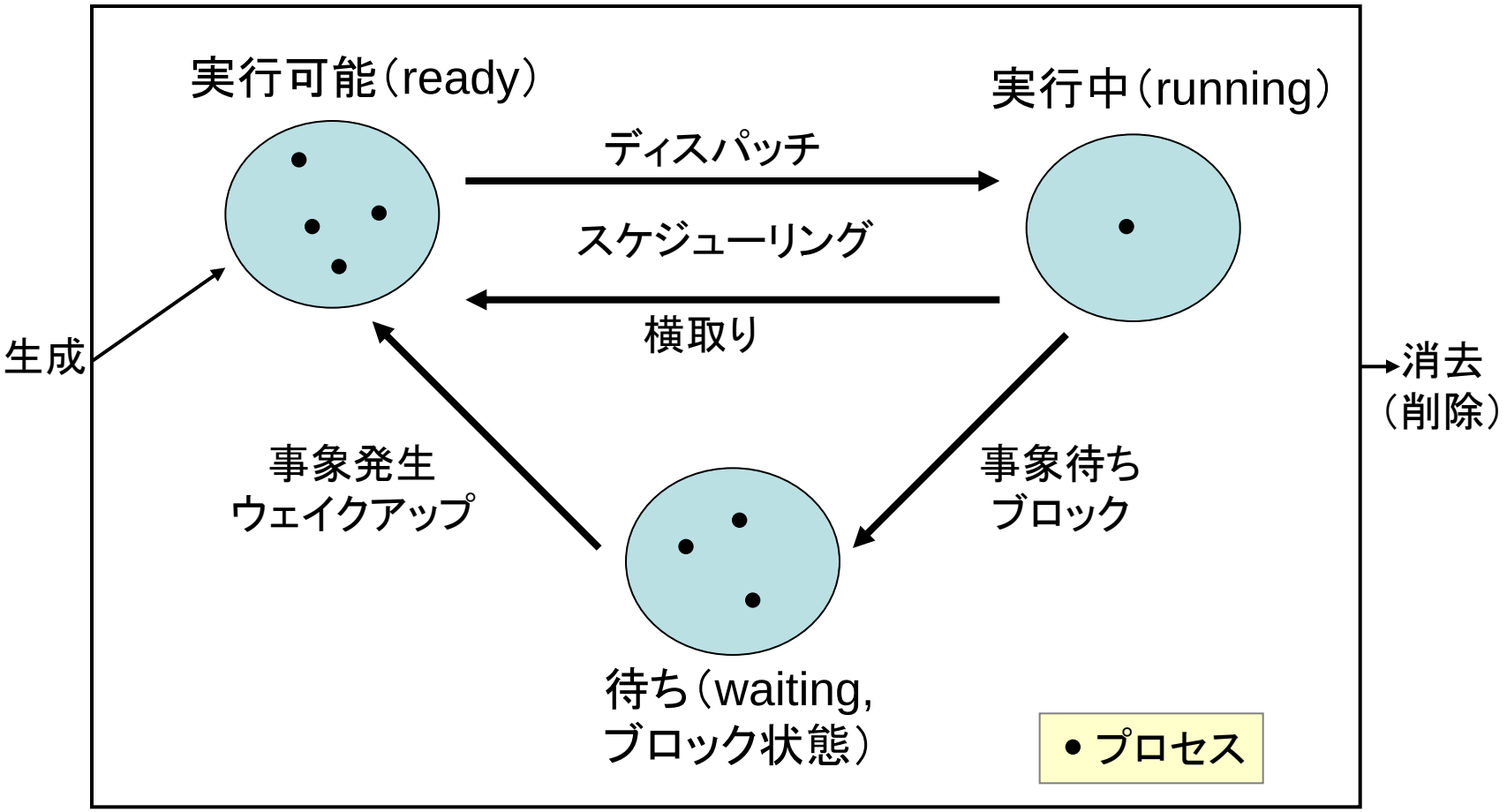
  grepの方がcatよりも処理が速いとき、

  grepをブロックしなければならない。

- OSが他のプロセスにCPUを割り当てているため停止 **（実行可能状態）**

その他に動作中の状態があるから、全体として ....

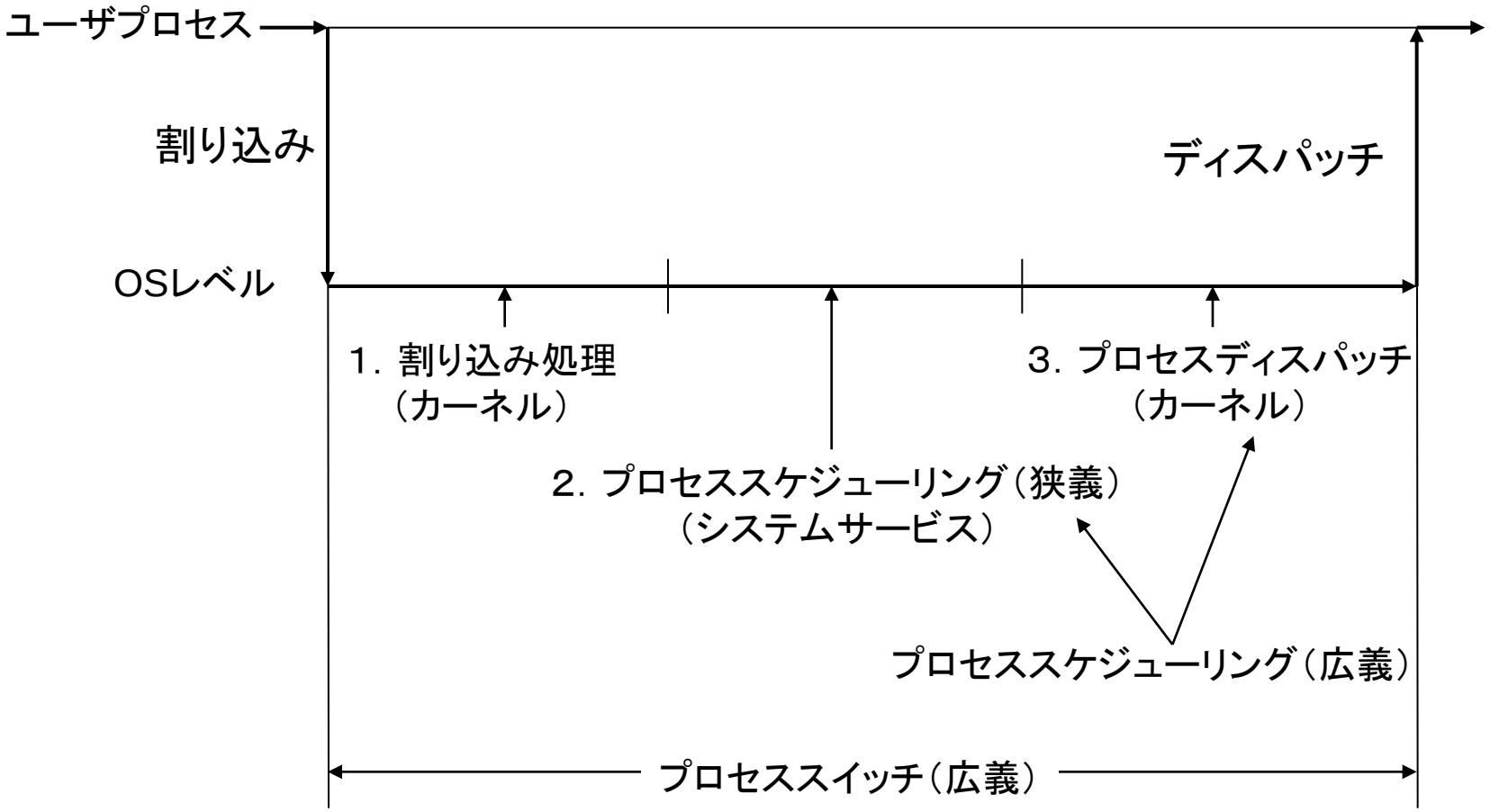
プロセスの状態と状態遷移図



「待ち」から「実行中」には直接遷移できない

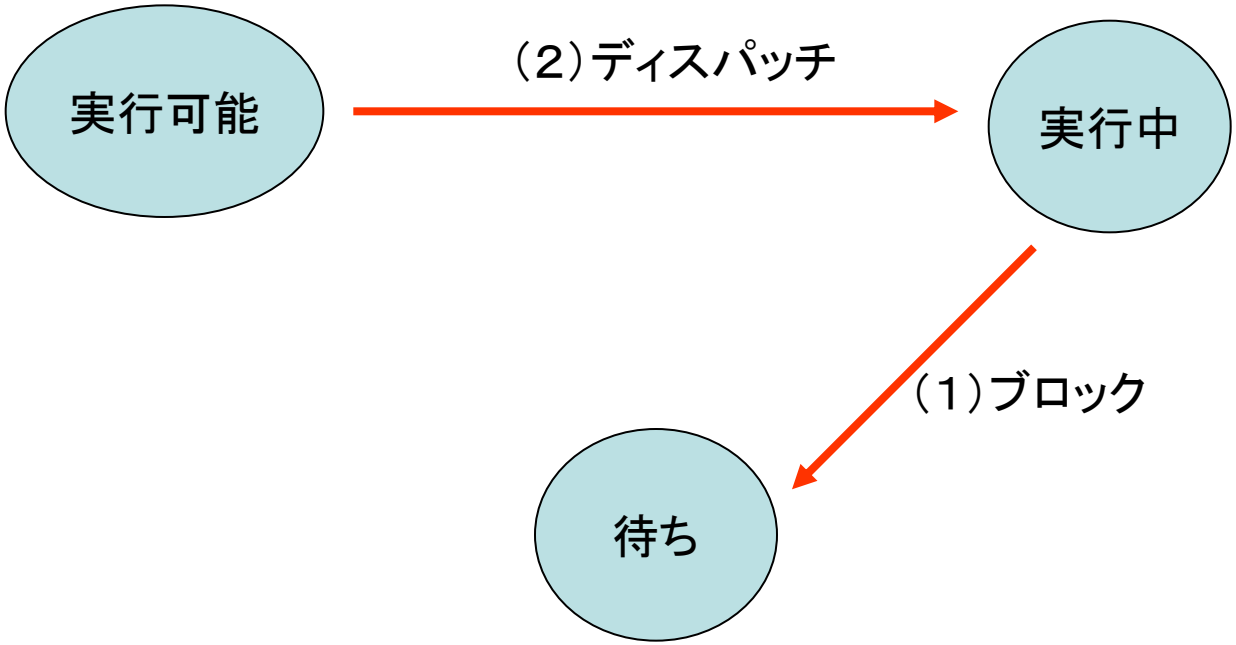
プロセス管理と割り込み処理

割り込みによるプロセス管理フロー



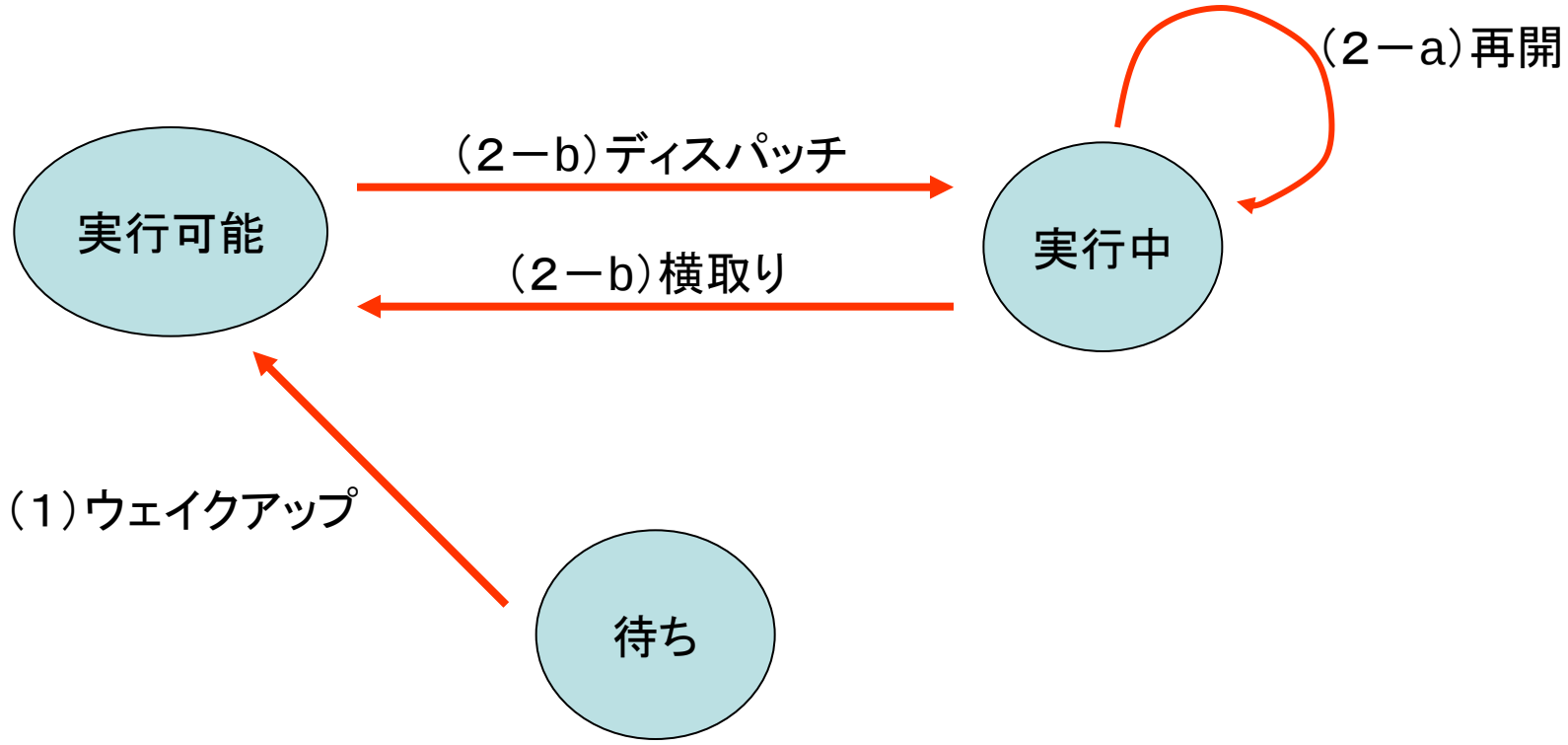
割り込みによるプロセス遷移

（A）ブロック割り込み
SVCなどの内部割り込みの場合



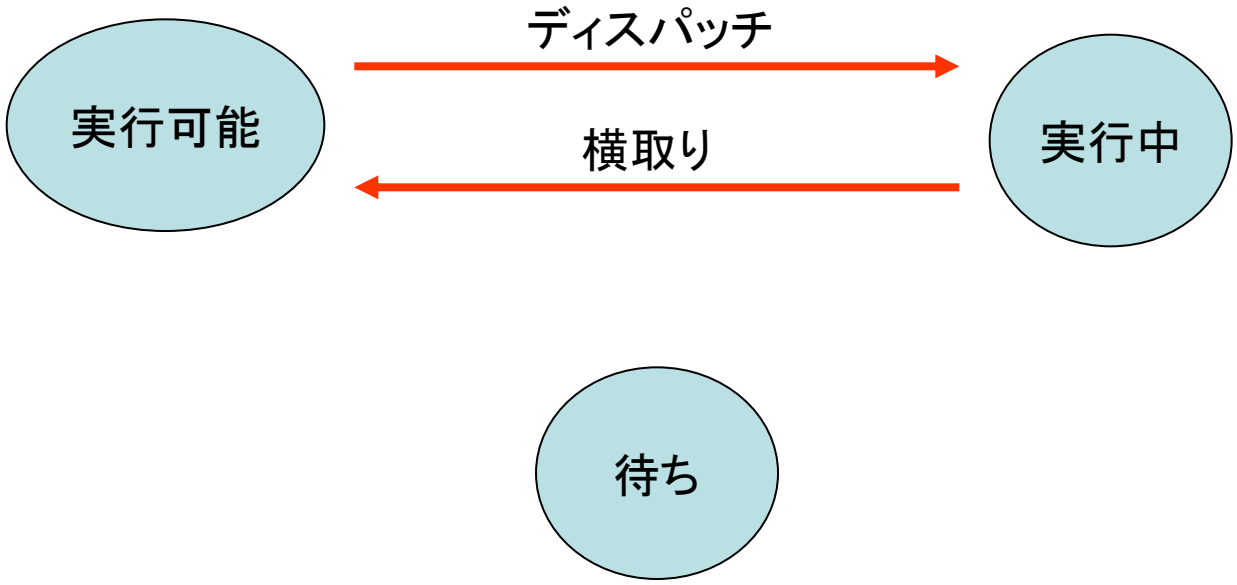
割り込みによるプロセス遷移

（B）ウェイクアップ割り込み
入出力割込みなどの待っていた事象の発生の場合



割り込みによるプロセス遷移

（C）タイマ割り込みによるプロセス
スケジューリング
ラウンドロビン方式など



ブロック要因とウェイクアップ要因の関係例

- 入出力処理を依頼するSVC（ブロック割り込み） ⇔ 入出力完了割り込み（ウェイクアップ割り込み）
- ページフォールト（命令実行例外）（ブロック割り込み） ⇔ ページを読み出す入出力完了割り込み（ウェイクアップ割り込み）
- プロセス間通信受信用SVC（ブロック割り込み） ⇔ プロセス間通信送信用SVC（ウェイクアップ割り込み）

演習クイズ（第二回）

- 割り込み要因において、内部割り込みおよび外部割り込みについて例示しながら説明せよ。

演習クイズ（第二回）

- 「マルチタスキング」を以下の言葉を使って説明せよ。
  - プロセス（タスク）、プロセッサ、時分割制御

演習クイズ（第二回）

プロセスの状態遷移図を描いて確認せよ。