オペレーティングシステム

第5回　同期･プロセス間通信

情報科学メジャー
鏑木崇史

排他制御の必要性

ある　資源R　が　スレッドT　に利用されている際に
ほかの　スレッドS　がその資源を利用したい場合を考える

クリティカルセクション

複数のプロセスがある共有メモリ領域を競合するアクセスをしているとき、
競合アクセスしているプログラム部分

相互排除

ある資源を同時に利用できるスレッド数が最大1の場合を相互排除
mutual exclusion (mutex / lock) と呼ぶ

Linuxのプロセス間通信

プロセス間通信

プロセスの間で行われるデータ交換のこと

共有メモリー

利点：アクセススピード

欠点：同時に書き込むと競合が発生する

→解決の機構はユーザ側で準備する

セマフォ

整数型のデータを親子関係の無いプロセス間で共有できる

→主に共有メモリの相互排他

マップドメモリー

複数のプロセスがファイルを介してデータを交換する

高速化のためにメモリ上にあるように仮想アドレス空間を用いる

Linuxのプロセス間通信

パイプ

親子関係にあるプロセス間の一方向の通信を実現

名前付きパイプ

親子関係がなくても全てのプロセスが
FIFOを作成、アクセス、削除できる

ソケット通信

依存関係のないプロセス間での通信を実現
他のマシンにあるプロセスと通信できる

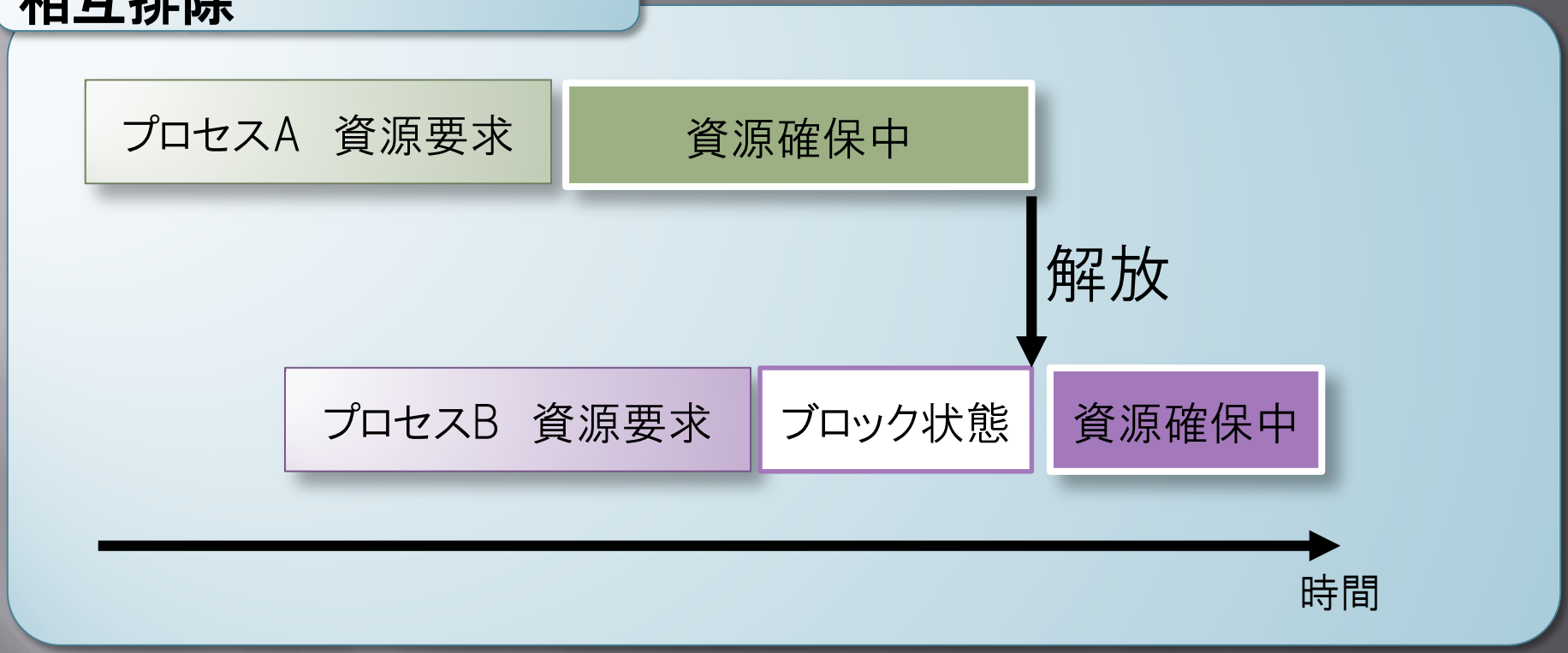
相互排除の条件

資源がどのスレッドにも利用されていない場合
→資源を利用しようとしたスレッドは直ちに資源が利用できる

複数のスレッドが資源を利用しようとしている場合
→資源利用できるスレッドの選択が無期限に延期されない
（デッドロック防止）

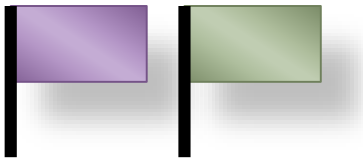
どのスレッドも資源を利用できる機会が公平にあたたえられる
（公平性）

相互排除



フラグ （Petersonアルゴリズム）

1. 待ち状態であることを示すフラグを「真」にする



待ち状態

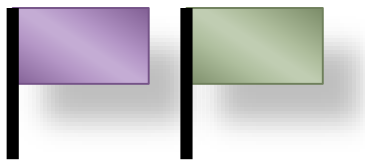
順番

プロセスA
クリティカルセッション
実行中

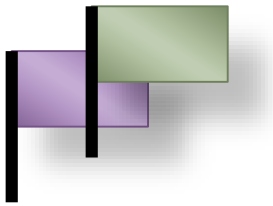
フラグ （Petersonアルゴリズム）

1. 待ち状態であることを示すフラグを「真」にする

2. 自分のプロセスIDを順番リストに記載する



待ち状態

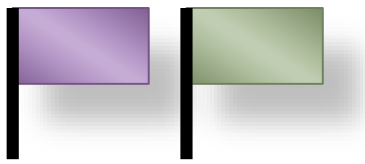


順番

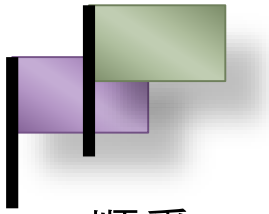
プロセスA
クリティカルセッション
実行中

フラグ （Petersonアルゴリズム）

1. 待ち状態であることを示すフラグを「真」にする
2. 自分のプロセスIDを順番リストに記載する
3. ほかのプロセスのフラグが「偽」になり、順番リストが自分のプロセスIDになるまで待つ。（繰り返し確認をする）



待ち状態



順番

プロセスA
クリティカルセッション
実行中

フラグ （Petersonアルゴリズム）

1. 待ち状態であることを示すフラグを「真」にする
2. 自分のプロセスIDを順番リストに記載する
3. ほかのプロセスのフラグが「偽」になり、順番リストが自分のプロセスIDになるまで待つ。（繰り返し確認をする）
4. 3. を満たしたらクリティカルセッションを実行する

待ち状態

順番

プロセスB
クリティカルセッション
実行中

フラグ （Petersonアルゴリズム）

1. 待ち状態であることを示すフラグを「真」にする
2. 自分のプロセスIDを順番リストに記載する
3. ほかのプロセスのフラグが「偽」になり、順番リストが自分のプロセスIDになるまで待つ。（繰り返し確認をする）
4. 3. を満たしたらクリティカルセッションを実行する
5. フラグを「偽」にする

待ち状態

順番

プロセスB
クリティカルセッション
実行中

フラグ （Petersonアルゴリズム）

1. 待ち状態であることを示すフラグを「真」にする
2. 自分のプロセスIDを順番リストに記載する
3. ほかのプロセスのフラグが「偽」になり、順番リストが自分のプロセスIDになるまで待つ。（繰り返し確認をする）
4. 3. を満たしたら
5. フラグを「偽」に

ビジーウェイト （スピンロック）

・CPUがアイドル状態にならない
・短時間で完了する場合は効果的なことも

待ち状態

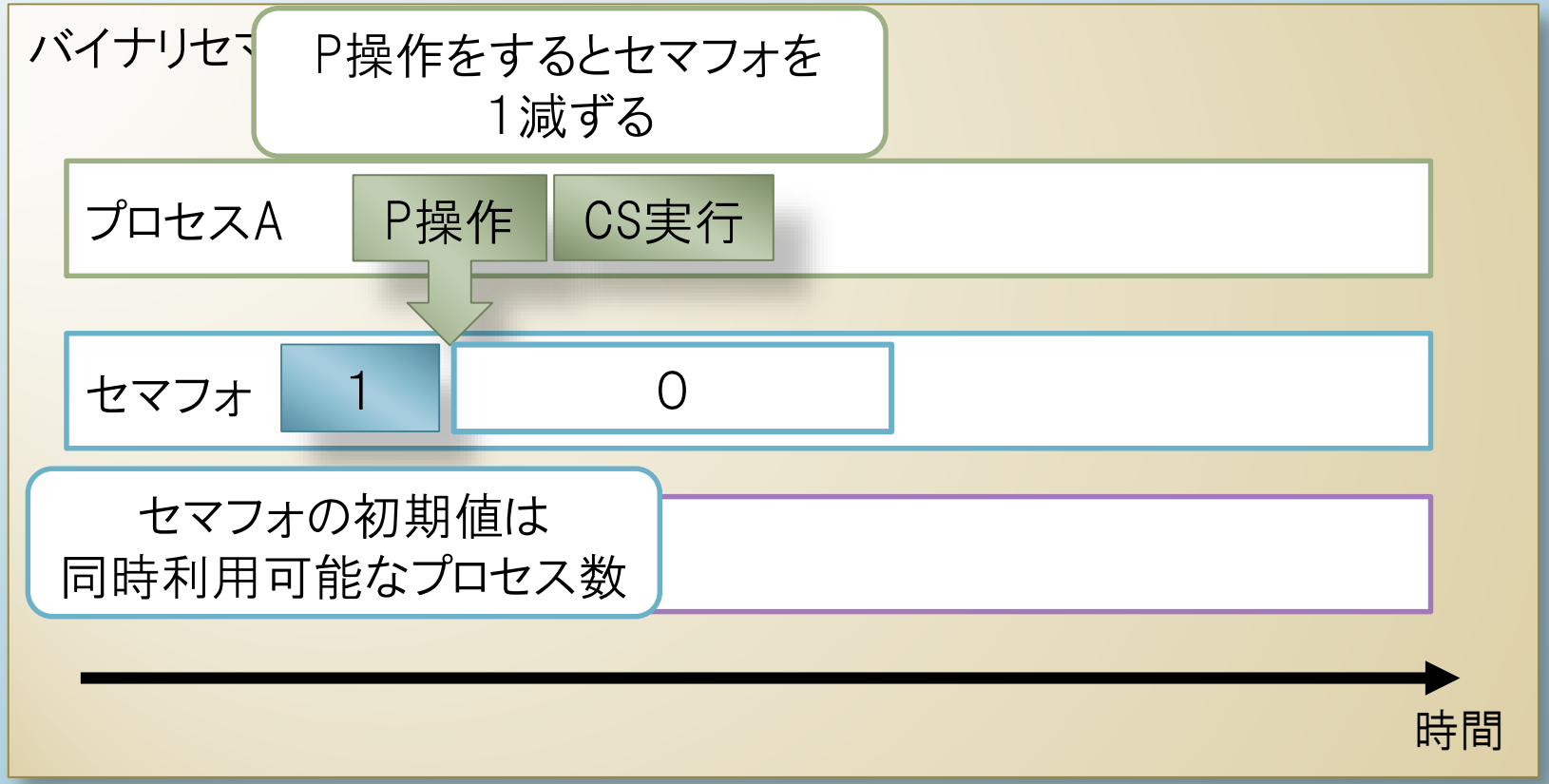
順番

プロセスB
クリティカルセッション
実行中

セマフォ

資源が何個使用可能かを示す記録する方式

任意個の資源を扱うセマフォ　→カウンティング（計数）セマフォ
値が0と1に制限されているセマフォ　→バイナリ（2進）セマフォ



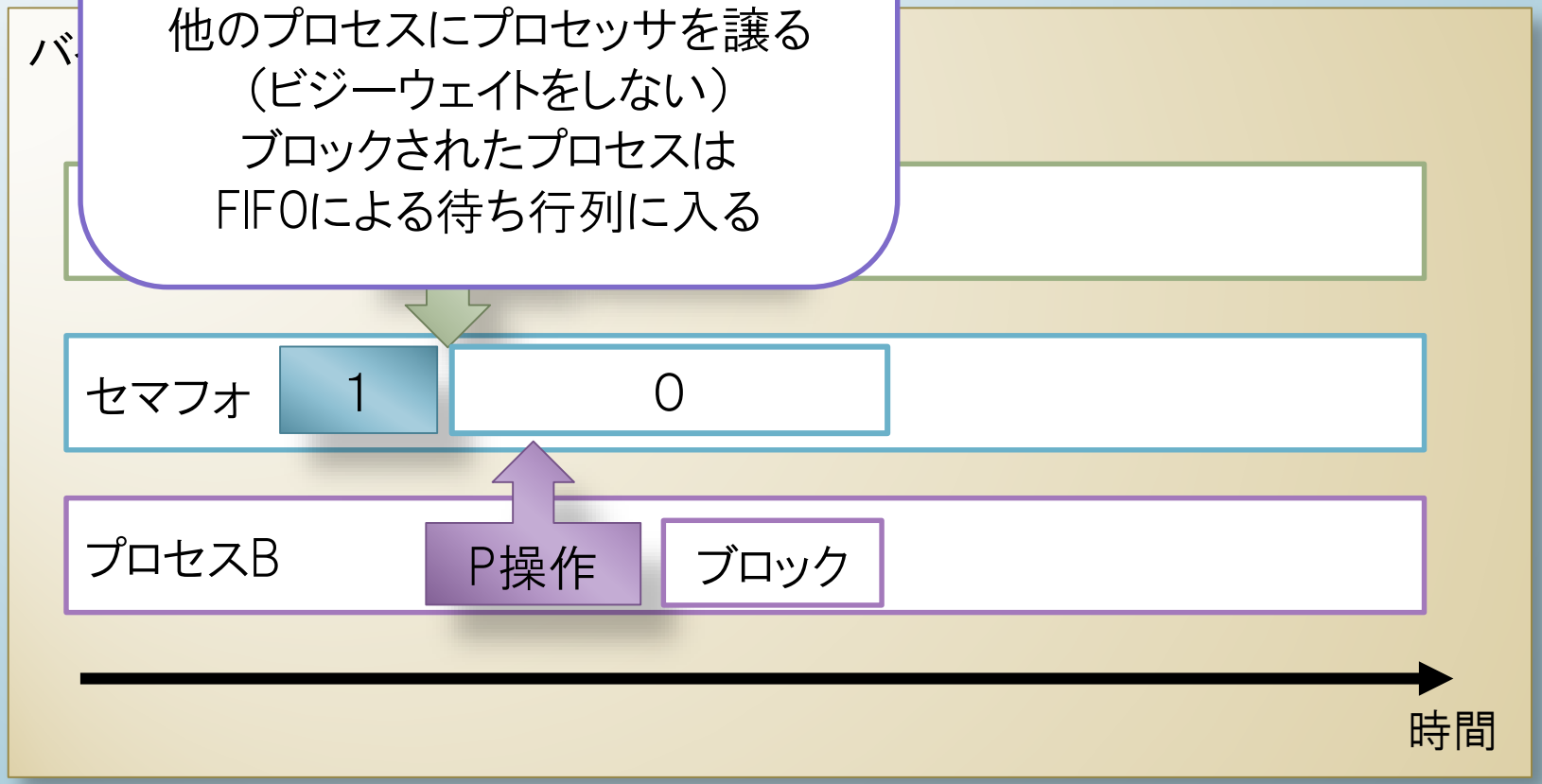
セマフォ

資源が何個使用可能かを示す記録する方式

任意個の資源を扱うセマフォ：カウンティング（計数）セマフォ

0と1のみ：バイナリ（2進）セマフォ

P操作の結果セマフォが
0を下回る場合はブロックされる
その際、実行を中断し、
他のプロセスにプロセッサを譲る
（ビジーウェイトをしない）
ブロックされたプロセスは
FIFOによる待ち行列に入る



セマフォ

資源が何個使用可能かを示す記録する方式

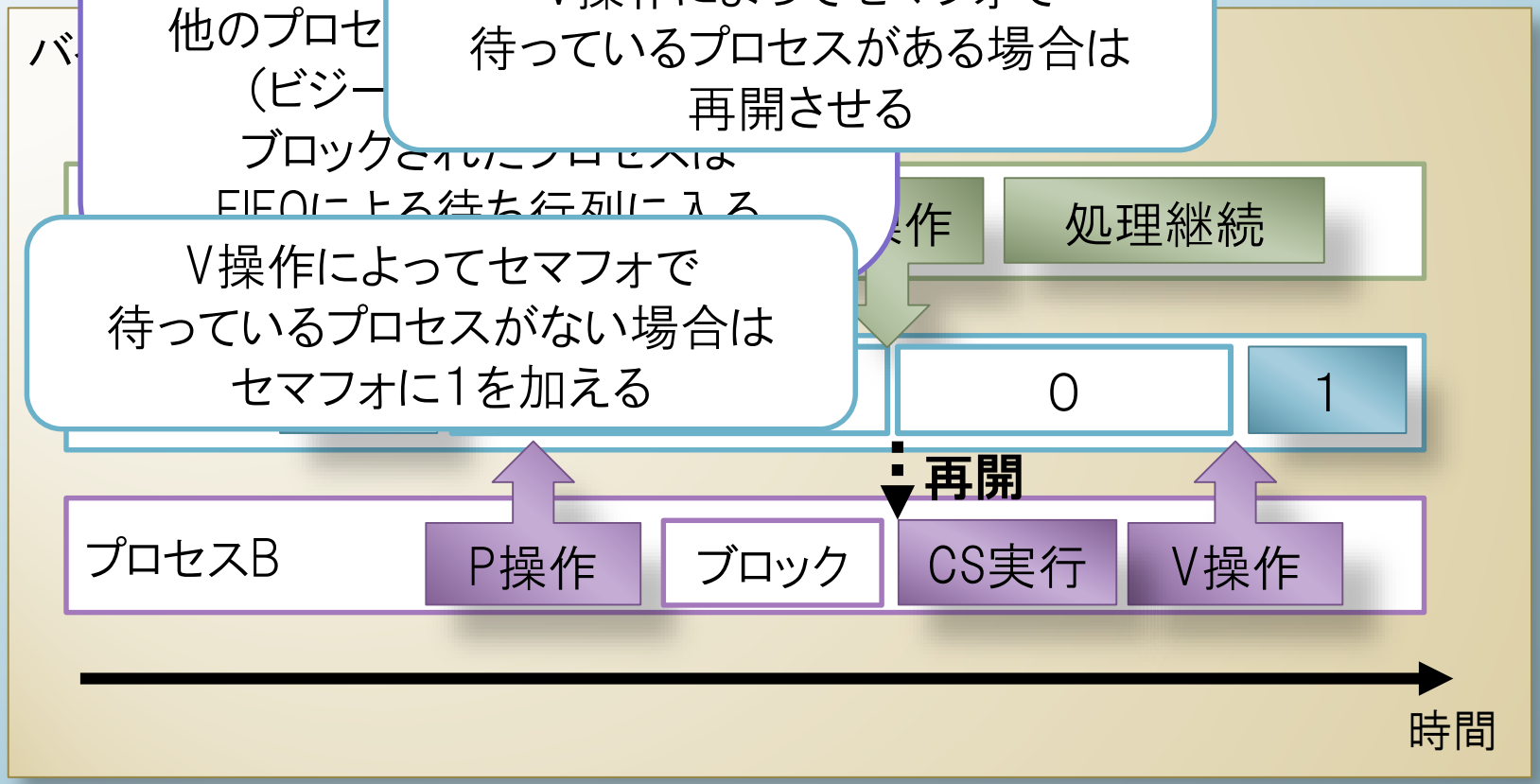
任意個の資源を扱うセマフォ：カウンティング（計数）セマフォ

2個の資源を扱うセマフォ：バイナリ（2進）セマフォ

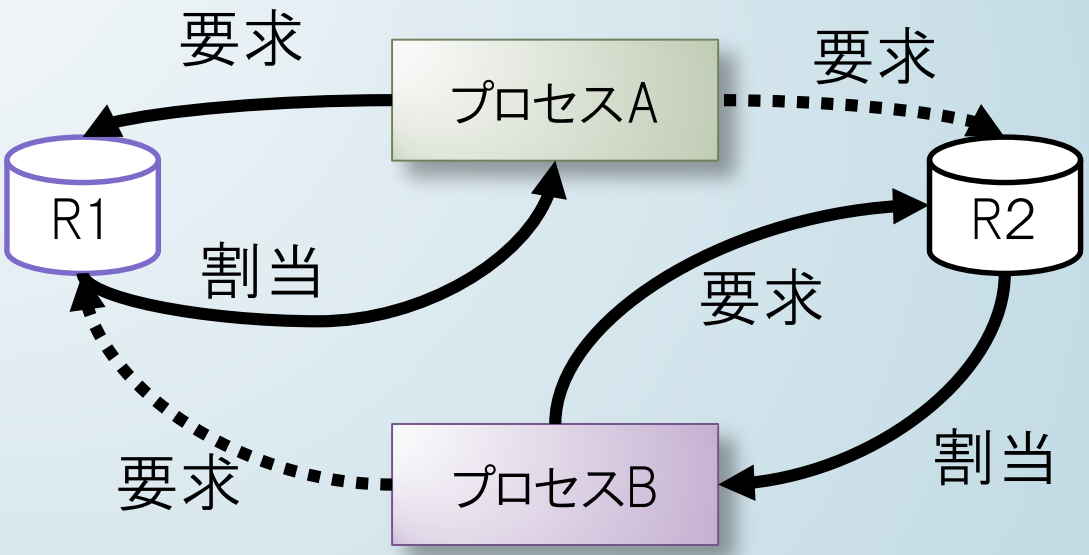
P操作の結果セマフォが
0を下回る場合はブロックされる
その際
他のプロセ
（ビジー
ブロックされたプロセスは
FIFOによる待ち行列に入る

V操作によってセマフォで
待っているプロセスがある場合は
再開させる

V操作によってセマフォで
待っているプロセスがない場合は
セマフォに1を加える



デッドロック



発生するための条件
(1)相互排除
(2)確保待ち
(3)横取り不可
(4)循環待ち

解決法
(1)防止
(2)回避
(3)回復
(4)検知
(5)無策

より強力

より非力

デッドロック解決法

防止

確保待ちにしない　→　必要な資源の要求を一括して要求する
循環待ちにしない　→　順序付けを行い、その順序に従い要求する

回避

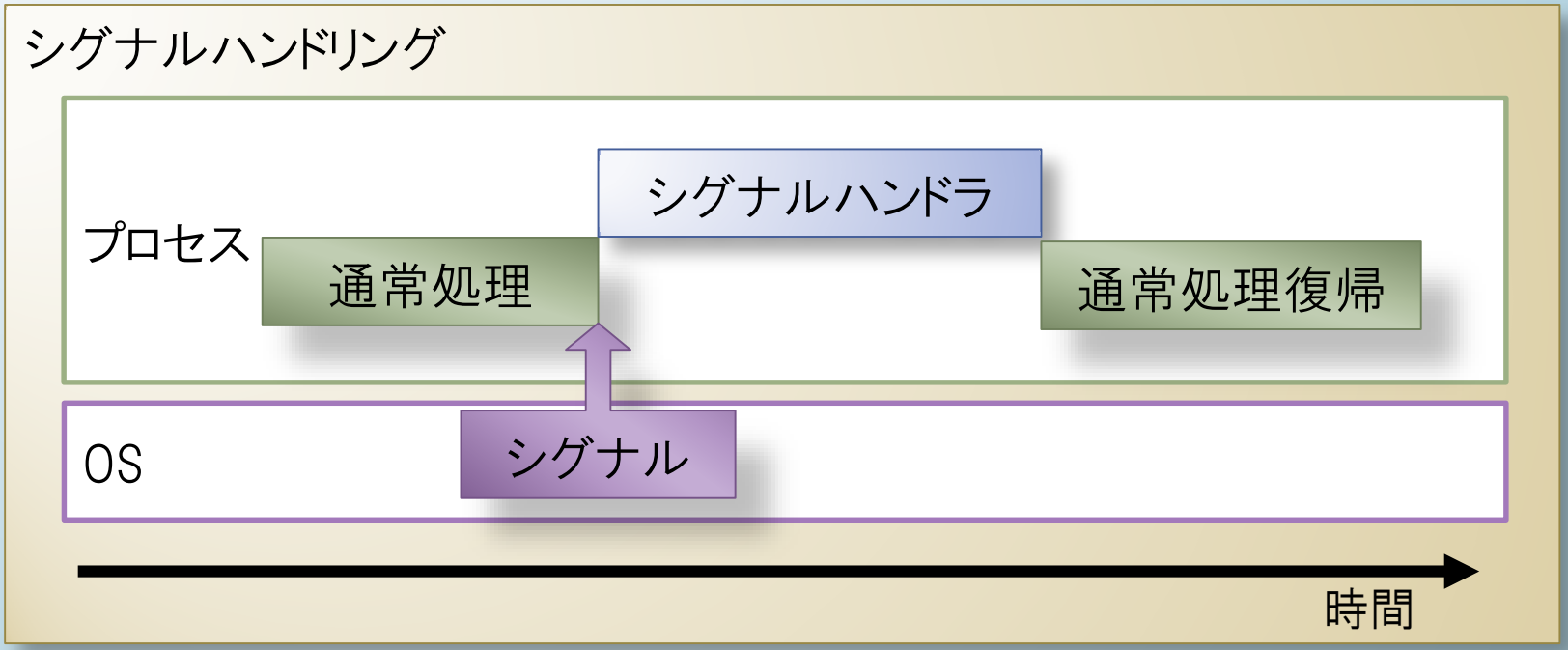
銀行家のアルゴリズム
→資源要求を満たしても、デッドロックが発生しない場合のみ割当
ただし、次の3種類の情報が必要となる
(1)すべてのプロセスが要求する可能性がある資源量
(2)各プロセスが占有している資源量
(3)システムに存在する資源量

回復

デッドロックを検出し、解消する方策
循環待ちを解消する
(1)強制終了させる
(2)ロールバック(後退復帰)させる　→チェックポイント･リスタート

Linuxのシグナル

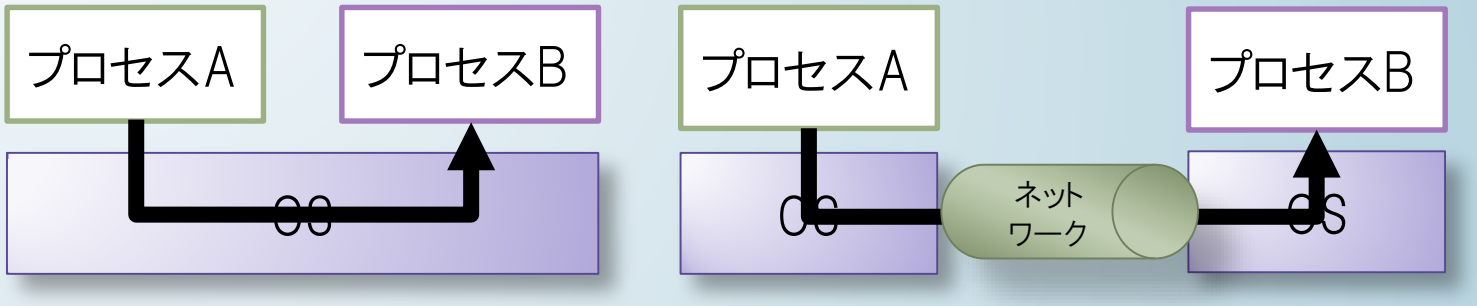
プロセスにシグナルを送信すると、そのプロセスの正常処理に
割り込んで、シグナル固有の処理(シグナルハンドラ) が実行される



通常はデフォルトのハンドラが設定されているが、独自のハンドラを
設定することも可能　→例)デーモンプロセスのSIGHUP

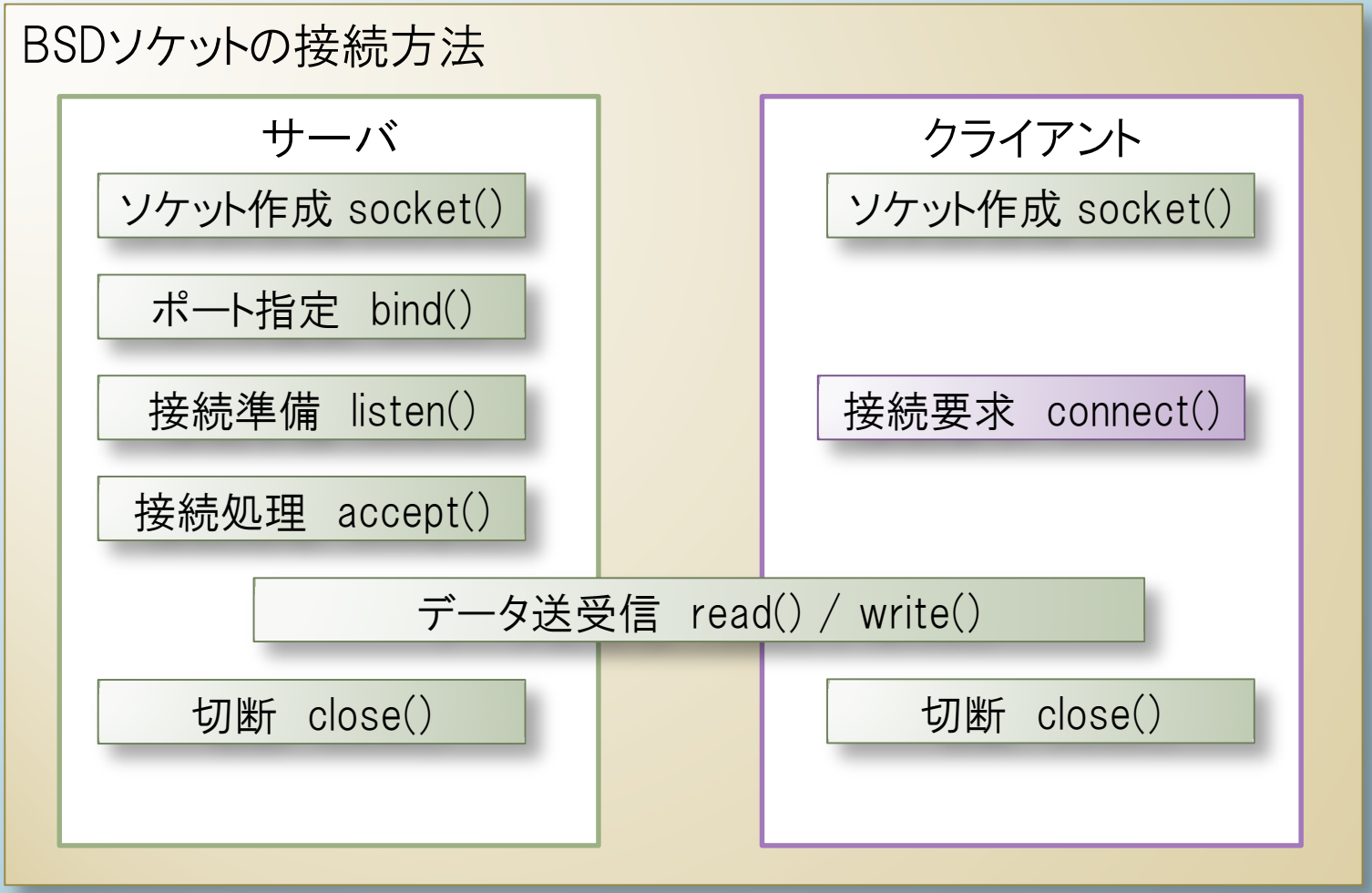
ソケット通信

主にネットワークを経由して通信することを目的としたプロセス間通信



ソケット通信

主にネットワークを経由して通信することを目的としたプロセス間通信



今週の課題について

今週の課題はボリュームが多いので、
任意に4題選択して実施してください。

余力のある人は全部やってもらっても構いません。

課題1

システムコールshmgetとシステムコマンドipcsについて調査し、報告せよ

課題2

Moodleの 05_shm1.c と 05_shm1.c をコンパイルし、
次の実験を実施して挙動について簡潔に解説せよ。

```
$ touch key_data.dat
$ cc -o shm1 05_shm1.c
$ cc -o shm2 05_shm2.c
$ ./shm1 &
segment id as 851977

$ ipcs
------ 共有メモリセグメント --------
キー      shmid      所有者   権限      バイト   nattch     状態
0x2a010333 851977     student    600        25600      1
$ ./shm2
$ fg
 <Enterを押した後に Ctrl+z >
$ ./shm2
$ fg
<Enterを押す >
```

課題3

システムコマンドmkfifoについて調査し、報告せよ

課題4

パイプと名前付きパイプについて、
次の実験を実施して挙動について簡潔に解説せよ。

```
$ dmesg
$ dmesg | less
$ mkfifo fifo.test
$ dmesg > test.txt
$ cat test.txt
$ dmesg > fifo.test
```

別のコマンドWindowで

```
$ cat fifo.test
```

課題5

システムコールsemopについて調査し、報告せよ

課題6

Moodleの smp1.c と smp2.c をコンパイルし、
次の実験を実施して挙動について簡潔に解説せよ。

```
$ cc -o smp1 smp1.c
$ cc -o smp2 smp2.c
$ ./smp1
```

別のコマンドwindowで

```
$ ./smp2
```

課題7

シグナル（ハンドラ）について、次の実験を実施して挙動について簡潔に解説せよ。

```
$ kill -l
$ cc -o signal signal.c
$ ./signal
start PID:213178
```

別のコマンドWindowで

```
$ kill -SIGABRT 213178
```

課題8

ソケット通信について、次の実験を実施して挙動について簡潔に解説せよ。

```
$ cc -o server server.c
$ ./server
```

別のコマンドWindowで

```
$ netstat -lt
tcp    0    0 0.0.0.0:22222      0.0.0.0:*        LISTEN
$ cc -o client client.c
$ ./client
```

オペレーティングシステム

第6回
プロセス間通信とソケット

情報科学メジャー
鏑木崇史