東邦大学

生命の科学で未来をつなぐ

基本的な記憶管理
と
その手法１

東邦大学

記憶管理とは
何をすること か

東邦大学

記憶管理の問題

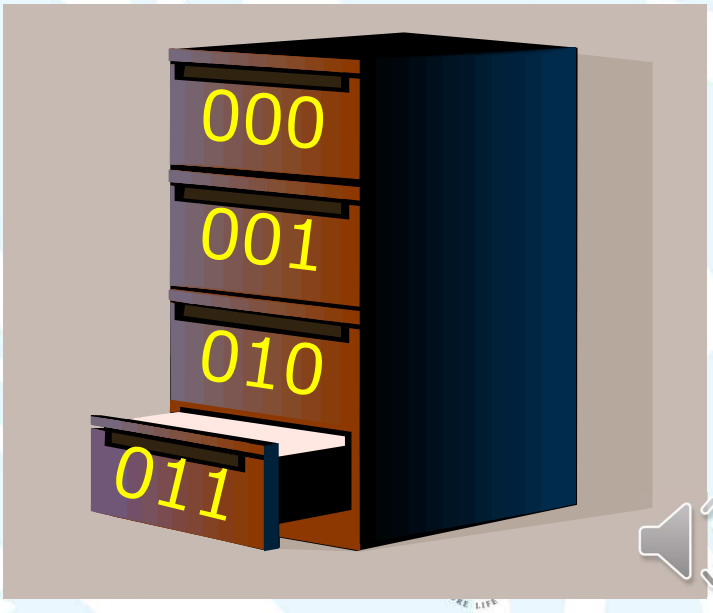
- メインメモリ（主記憶）をどう使うか？

記憶管理の問題

- メインメモリ（主記憶）をどう使うか？
  - （ディスクスペースの管理も似たような話があるが、さしあたってメインメモリだけについて考えよう）

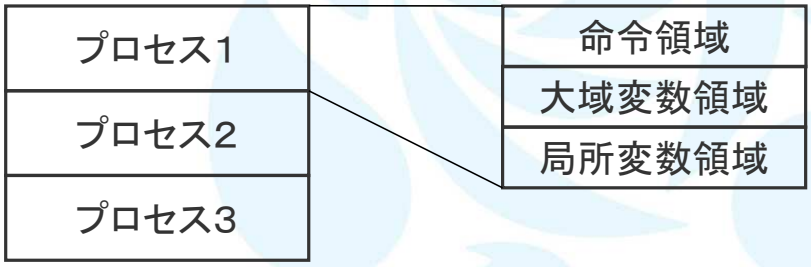
記憶管理の問題

- メインメモリ（主記憶）をどう使うか？
- メモリは、単純な１次元のスペース
  ０番地から
  （最大番地）まで、
  番号の付いた引出し



記憶管理の問題

- メインメモリ（主記憶）をどう使うか？
- メモリは、単純な１次元のスペース
  ０番地から (最大番地) まで番号の付いた引出し
- これを区画に分けて使いたい
  - 複数のプロセスを同時にメモリ上に置きたい
  - 同じプロセス内でも用途の違う領域に分けたい



記憶管理の問題

- メインメモリ（主記憶）をどう使うか？
- メモリは、単純な１次元のスペース
  ０番地から (最大番地) まで番号の付いた引出し
- これを区画に分けて使いたい
  - 複数のプロセスを同時にメモリ上に置きたい
  - 同じプロセス内でも用途の違う領域に分けたい
- しかも、スペースを有効に使いたい
  - 使っていない部分は追出して空けたい

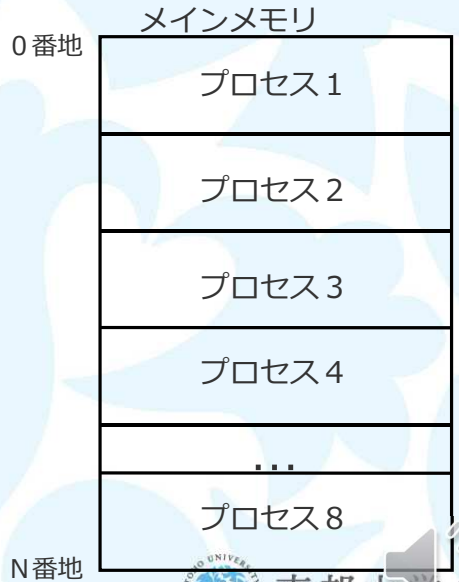
記憶管理の問題

- メインメモリ（主記憶）をどう使うか？
- メモリは、単純な１次元のスペース
  ０番地から (最大番地) まで番号の付いた引出し
- これを区画に分けて使いたい
  - 複数のプロセスを同時にメモリ上に置きたい
  - 同じプロセス内でも用途の違う領域に分けたい
- しかも、スペースを有効に使いたい
  - 使っていない部分は追出して空けたい

といった問題をうまく処理したい

区画に分ける問題

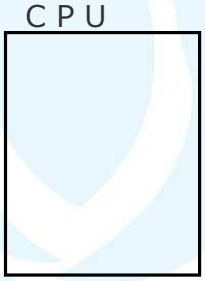
- たとえば、
  - プロセスを同時に８つ置きたい（多重プロセス）

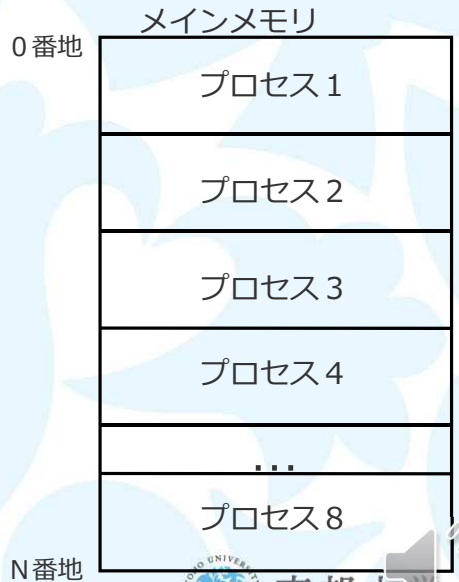


区画に分ける問題

- たとえば、
  - プロセスを同時に８つ置きたい（多重プロセス）

CPUは時分割で
プロセスを切替えるが





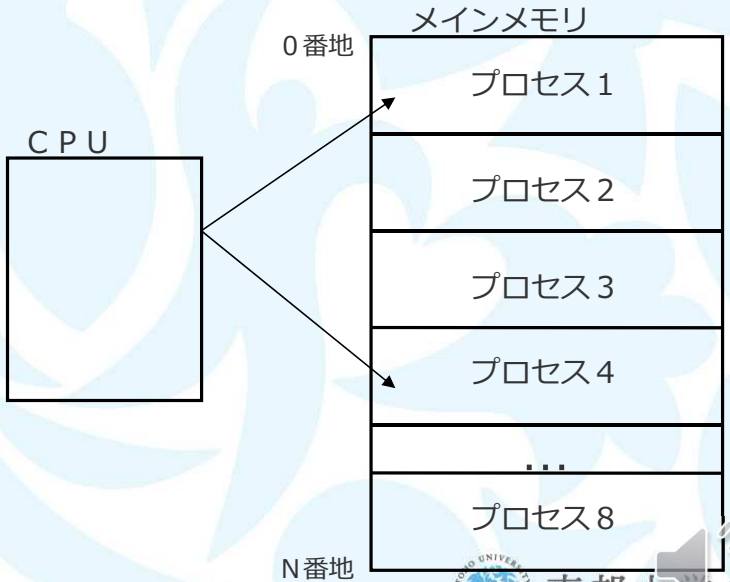
区画に分ける問題

- たとえば、
  - プロセスを同時に８つ置きたい（多重プロセス）

CPUは時分割で
プロセスを切替えるが

メモリはプロセスを並べて
置いておき、アドレスの
違いで場所を区別する

「プロセス１つ分の空間で
入替えて使う」のではない



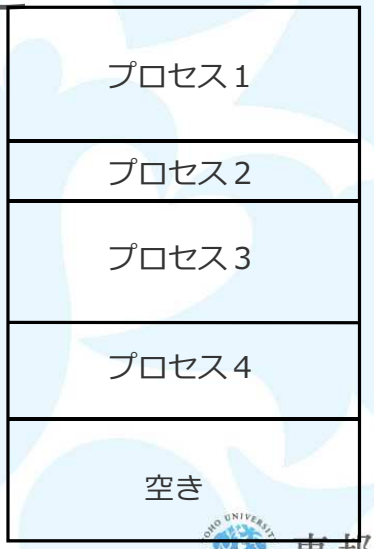
複数のプロセスを置くとき何が問題か？

複数のプロセスを置くとき何が問題か？

- プロセスの大きさ（メモリ必要量）は
  プロセスによって異なる
  - 大きいプロセス（命令が沢山、データが沢山）
    も小さいプロセスもある

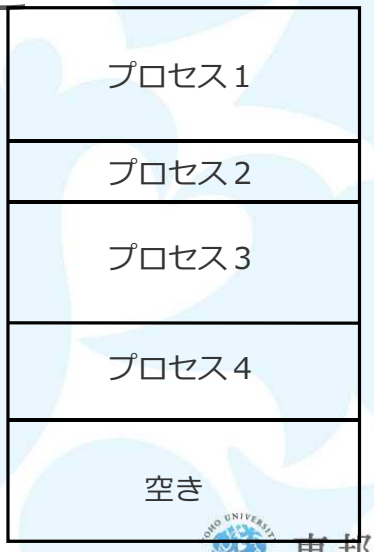
複数のプロセスを置くとき何が問題か？

- プロセスの大きさ（メモリ必要量）は
  プロセスによって異なる
- 単純に、端から詰めて置くと



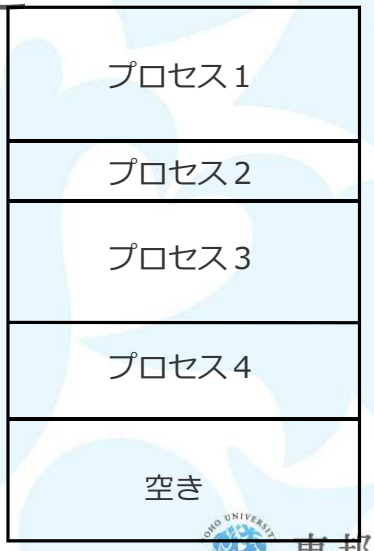
複数のプロセスを置くとき何が問題か？

- プロセスの大きさ（メモリ必要量）は
  プロセスによって異なる
- 単純に、端から詰めて置くと
  - どこが誰に使われているか
    表を作って管理する
  - 各区画の先頭番地と長さを
    管理表に記憶しておく



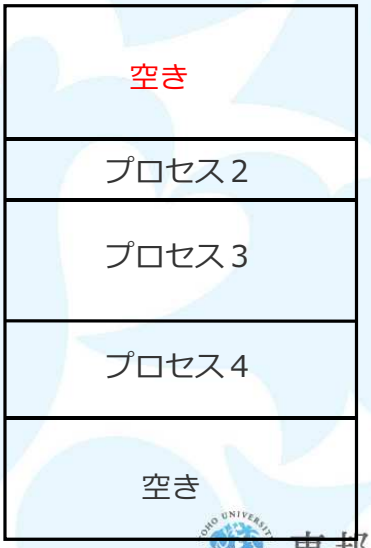
複数のプロセスを置くとき何が問題か？

- プロセスの大きさ（メモリ必要量）は
  プロセスによって異なる
- 単純に、端から詰めて置くと
  - どこが誰に使われているか
    表を作って管理する
  - 各区画の先頭番地と長さを
    管理表に記憶しておく
  - 空き地も管理しておくと便利



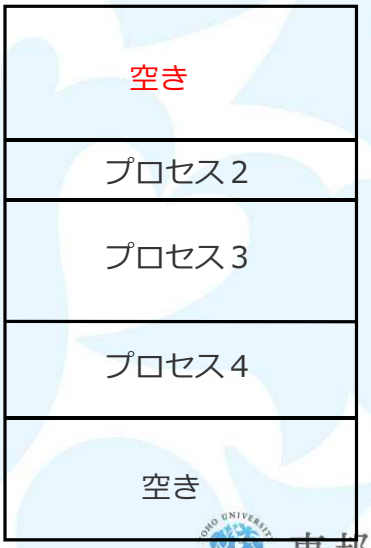
複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる



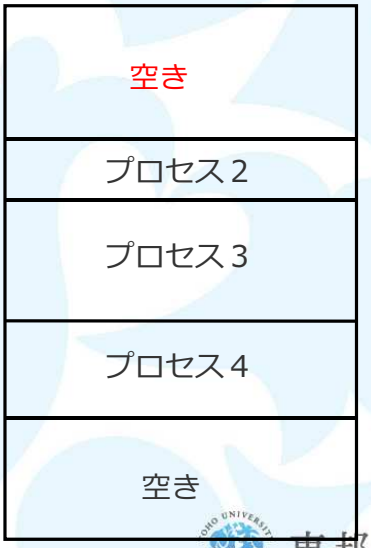
複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる
    ⇒ 空き地が飛び飛び ⇒ 管理表の工夫が必要



複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる
    ⇒ 空き地が飛び飛び ⇒ 管理表の工夫が必要
- 新しいプロセス５が誕生
  してスペースが欲しい



複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる
    ⇒ 空き地が飛び飛び ⇒ 管理表の工夫が必要
- 新しいプロセス５が誕生
  してスペースが欲しい
  - ５は１より少し小さいので
    １の空き地を使う

複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる
    ⇒ 空き地が飛び飛び ⇒ 管理表の工夫が必要
- 新しいプロセス５が誕生してスペースが欲しい
  - ５は１より少し小さいので１の空き地を使う
  - 次に３が終了

複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる
    ⇒ 空き地が飛び飛び ⇒ 管理表の工夫が必要
- 新しいプロセス５が誕生してスペースが欲しい
  - ５は１より少し小さいので１の空き地を使う
  - 次に３が終了・６が誕生

複数のプロセスを置くとき何が問題か？

- プロセス１が終了したときを考える
  - プロセス１のスペースは空き地になる
    ⇒ 空き地が飛び飛び ⇒ 管理表の工夫が必要
- 新しいプロセス５が誕生してスペースが欲しい
  - ５は１より少し小さいので１の空き地を使う
  - 次に３が終了・６が誕生

「空き地にちょっと小さい新プロセスを入れる」を繰り返すと、小さい空き地が飛び飛びに残る

複数のプロセスを置くとき何が問題か？

- 領域確保・解放を繰り返した結果
  飛び飛びの空き地がたくさんできる

複数のプロセスを置くとき何が問題か？

- 領域確保・解放を繰り返した結果
  飛び飛びの空き地がたくさんできる
  ↓
- この空き地は連続していないので
  まとめて１つのプロセスに割り当てる
  ことが出来ない　⇒　結局使えない

複数のプロセスを置くとき何が問題か？

- 領域確保・解放を繰り返した結果
  飛び飛びの空き地がたくさんできる
  ↓
- この空き地は連続していないので
  まとめて１つのプロセスに割り当てる
  ことが出来ない　⇒　結局使えない
  ↓
- 無駄なスペースが増え続ける（食い尽くす）

複数のプロセスを置くとき何が問題か？

- 領域確保・解放を繰り返した結果
  飛び飛びの空き地がたくさんできる
  ↓
- この空き地は連続していないので
  まとめて１つのプロセスに割り当てる
  ことが出来ない　⇒　結局使えない
  ↓
- 無駄なスペースが増え続ける（食い尽くす）

（外部）フラグメンテーション　と呼ぶ

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）
- プロセス１と５の大きさの
  差が空き地になる

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）
- プロセス１と５の大きさの
  差が空き地になる
  ⇒ すべてのプロセスが同じ
  大きさなら空かない

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）
- プロセス１と５の大きさの差が空き地になる
  ⇒ すべてのプロセスが同じ大きさなら空かない
  （「固定長区画割当て」なら、すきまが出来ない）

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）
- プロセス１と５の大きさの差が空き地になる
  ⇒ すべてのプロセスが同じ大きさなら空かない
  （「固定長区画割当て」なら、すきまが出来ない）
- 空き地が飛び飛びだから小さくて再利用不可
  ⇒ くっつけて大きくまとめれば再利用できる

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）
- プロセス１と５の大きさの差が空き地になる
  ⇒ すべてのプロセスが同じ大きさなら空かない
  （「固定長区画割当て」なら、すきまが出来ない）
- 空き地が飛び飛びだから小さくて再利用不可
  ⇒ くっつけて大きくまとめれば再利用できる
  （コンパクション・ガベージコレクション）

なぜ小さい空き地が出来るのか？

- 領域確保・解放を繰り返すとき、
  飛び飛びの空き地ができるのはなぜか？
  （どういう条件のときか？）
- プロセス１と５の大きさの差が空き地になる
  ⇒ すべてのプロセスが同じ大きさなら空かない
  （「固定長区画割当て」なら、すきまが出来ない）
- 空き地が飛び飛びだから小さくて再利用不可
  ⇒ くっつけて大きくまとめれば再利用できる
  （コンパクション・ガベージコレクション）
  ⇒ 処理時間が大きく、ここでは使い物にならない

ここまでのまとめ

- メモリ上にプロセスを並べて置く
- その結果、スペースの割当て管理が必要
- プロセスの大きさに合わせて（可変長）割当てると
  生成・消滅（＝確保・解放）を繰り返すうち
  切れ切れの無駄な空き地が溜まる
  （外部）フラグメンテーションと呼ぶ
- （外部）フラグメンテーションの起きる要因は
  ①可変長割当て（長さいろいろ）であることと
  ②くっつけられない（移動できない）こと

記憶管理の考え方が掴めましたか？

○ ×

次へ