


# オペレーティングシステム

## 第6回 記憶領域管理

情報科学メジャー  
鍋木崇史



プログラム

メモリを1つのプログラムに  
占有させるのは資源の観点から  
望ましくない



# アドレス



複数のプログラムを並行して実行する

メモリを必要に応じて分割する

必要なメモリの量はプログラムによって異なる  
プログラムは動的に生成される  
プログラムは動的に終了する

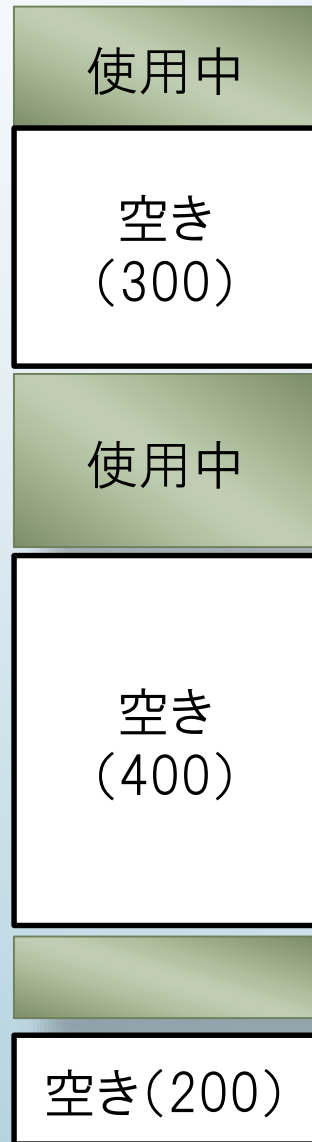
OSはメモリを動的に複数の可変長で割り当てる  
記憶領域管理が必要



## 可変長領域管理

(外部)断片化

→利用されていない領域が分散している状態



新規確保(200)



先頭適合

→空き領域の中で最初に  
みつかった領域を利用

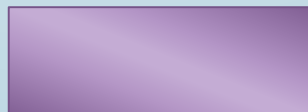
新規確保(200)



最悪適合

→空き領域の中で要求を満たす  
最大の空き領域を利用

新規確保(200)

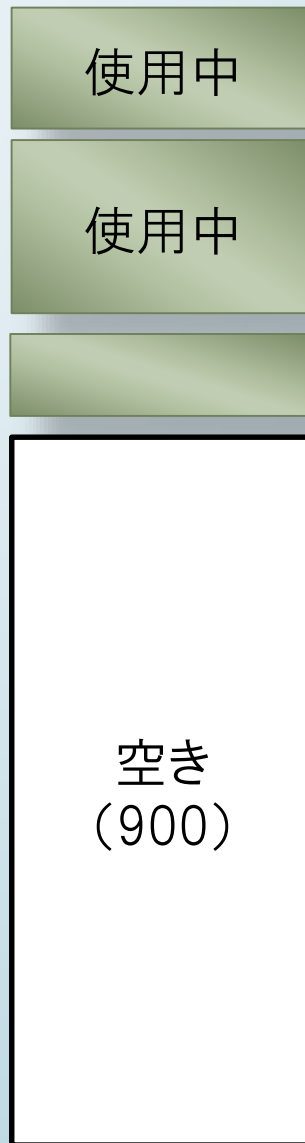
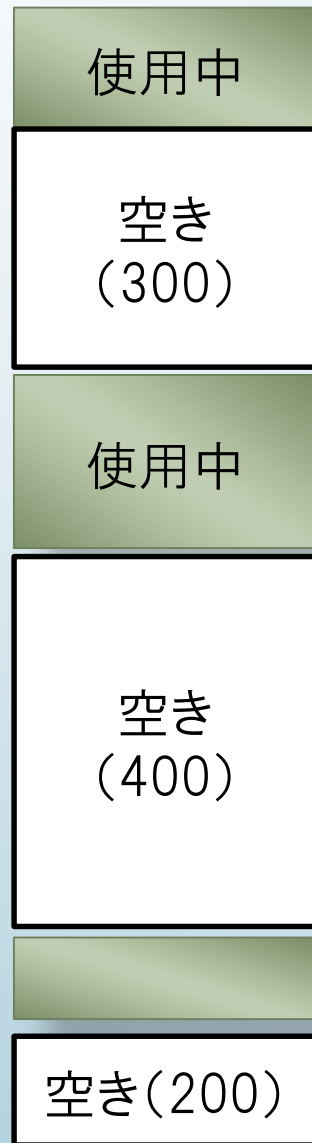


最良適合

→空き領域の中で要求を満たす  
最小の空き領域を利用



## コンパクション



使用されている領域を移動して詰まった状態にすること

外部断片化を解消する有効な手段

(問題点)

領域移動に必要なデータコピーに時間がかかる

移動中はプログラムの通常処理を中断する必要がある

使用中のデータやプログラムが特定のアドレスに依存しないか確認する必要がある

→再配置可能(リロケータブル)



論理  
アドレス空間

物理  
アドレス空間

物理アドレス空間とは別に  
論理アドレス空間を導入する



# ページング

論理 ページ番号	論理 アドレス空間
0	a
1	b
2	c
3	d
4	
5	
6	
7	

それぞれのアドレス空間を  
ページとして分割する

プログラムが利用するメモリは  
論理空間では連続  
物理空間では不連続でもOK

物理 ページ番号	物理 アドレス空間
0	
1	
2	
3	b
4	c
5	
6	
7	
8	a
9	
10	d
...	



# ページング

論理 ページ番号	論理 アドレス空間
0	a
1	b
2	c
3	d
4	
5	
6	
7	

論理 ページ番号	物理 ページ番号
0	8
1	3
2	4
3	10
4	
5	
6	
7	

物理 ページ番号	物理 アドレス空間
0	
1	
2	
3	b
4	c
5	
6	
7	
8	a
9	
10	d
...	

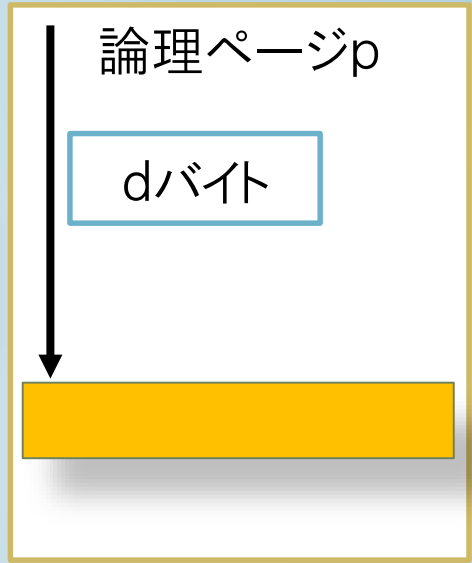
論理アドレス→物理アドレスへの  
ページテーブルを導入する

CPU内のメモリ管理ユニット(MMU)  
で実現される





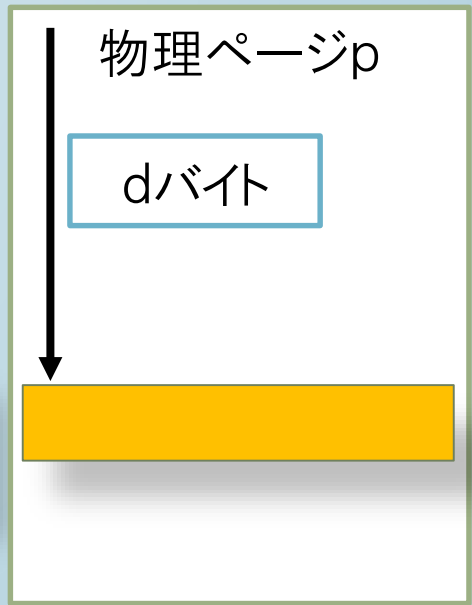
# ページング



論理 ページ番号	物理 ページ番号
0	8
1	3
2	4
3	10
4	
5	
6	
7	

論理ページ番号p

物理ページ番号t



## 論理アドレスを用いるメリット

物理メモリ以上のメモリ容量を扱える（ように見せかける）  
ハードディスクなどに主記憶の一部を退避させることで  
物理メモリ以上のメモリ容量を利用できる

メモリの割り当て

再配置可能性を考える必要がなく、外部断片化もない

保護機構

プロセスは独立した論理アドレス空間を持つので、  
複数のプロセスが主記憶を安全に分割して利用可能



# ページングの問題と解決策

## ページサイズの大きさ

小さい → ページテーブルの肥大化

大きい → (内部)断片化による領域の無駄が発生する

## 多段ページテーブル

論理アドレス空間が大きくなっても実際に利用する領域が小さい場合、ページテーブル階層化することで節約する

## アドレス変換キャッシュ

ページテーブルは物理メモリ上にあるため、頻繁にアクセスする領域を一時保存するtranslation look aside buffer (TLB)を置く。  
TLBにエントリがある場合をHit ない場合をMissと呼ぶ



# ページ保護

ページテーブルのエントリにページ保護情報(メタデータ)を付与することでページ保護やページのアクセス履歴を保存する

## 存在ビット

主記憶にページが存在するかを示す

→ 存在しない場合は **ページフォルト** が発生

## ページ保護 →【課題1】

コード部分 → 実行可能 書き込み禁止

データ部分 → 実行不可 書き込み可

→ ページ保護に反するアクセスは **アクセス違反例外** が発生

## アクセス履歴

参照ビット → 読み込みが行われたページ

変更ビット → 書き込みが行われたページ(Dirty bit)



## 課題1

man procとして/proc/[pid]/mapsの説明を読み、ページ保護がどのように表示されるかを簡潔に説明せよ。

## 課題2

Moodleの read.c をコンパイルしどのアドレスまでエラーなくアクセスできるか試して報告せよ。

なぜ範囲外のアドレスにアクセスしてもすぐにはエラーにならないか簡潔に解説せよ。

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



## 課題3

ページの確保について、Moodleのmmap.cを持ちいてその挙動を解説せよ。

```
$ cc -o mmap mmap.c
$ ./mmap
*** memory map before memory allocation ***
....
*** succeeded to allocate memory: address = 0x7fe1f52f8000; size = 0x6400000 ***
....
```

