UNIXシステムプログラミング

第1回 イントロダクション & 課題1

2022年4月12日

情報工学科

寺岡文男

コンピュータのOSについて

- PCは電源OFFの状態
- 起動するとOS (Windows or Linux)選択画面が表示
- Linuxを選択し，Enterキーを押して決定

- 選択画面で10秒間操作しないとWindowsが起動
  → 再起動して再度OS選択画面を表示させLinuxを起動

概要

- 2006年度設置の科目(選択科目)
  - 2019年度までは3年秋学期
- 対象とする受講生
  - “プログラミング第3同演習” の内容を理解している学生
    - 特に，ポインタ，構造体の理解は必須
  - “オペレーティングシステム” を受講していることが望ましい
- ねらい
  - アルゴリズムとデータ構造を理解する
  - UNIXのシステムコールを使いこなせるようになる
  - その結果として，システムプログラミングができるようになる
- 評価
  - 5題の課題により評価 (期末試験なし)

講義資料 & 講義の進め方

- CANVASを利用
  - 講義資料配付，テキスト配布，課題提出
- テキスト
  - 書き下ろし
  - 見直してはいるが，間違いを見つけたら知らせてください．
- 質問用ML
  - sysprog@inl.ics.keio.ac.jp

- TA 2名
- 講義のみ or 講義と簡単な演習

講義スケジュール (予定)

- 4/12: イントロ
  - 課題1 : Dijkstra
- 4/19: バッファキャッシュ1
- 4/26: バッファキャッシュ2
  - 課題2 : バッファキャッシュ
- 5/3: 文字と文字列操作
- 5/10: ファイル操作
- 5/17: シェル 1
- 5/24: シェル 2
  - 課題3 : mysh
- 5/31: ネットワーク 1
- 6/7: ネットワーク 2
- 6/14: クライアント・サーバ 1
  - 課題4 : myDHCP
- 6/21: クライアント・サーバ2
- 6/28: クライアント・サーバ3
  - 課題5 : myFTP
- 7/5, 12 予備

課題スケジュール

- 4/12 課題1 : ダイクストラのShortest Path First
  - 締切: 4/25 20:00 (2週間)
- 4/26 課題2 : OSのバッファキャッシュ管理
  - 締切: 5/23 20:00 (4週間)
- 5/24 課題3 : mysh (シェルの作成)
  - 締切: 6/20 20:00 (4週間)
- 6/14 課題4 : myDHCP (疑似DHCPの作成)
  - 締切: 7/11 20:00 (4週間)
- 6/28 課題5 : myFTP (疑似FTPの作成)
  - 締切: 7/25 20:00 (4週間)

課題とそのねらい (1)

- 課題1: DijkstraのSPF (Shortest Path First)アルゴリズム
  - 明示されているアルゴリズムからプログラムを作成する.
  - 指定されている仕様に従う

- 課題2: OSのバッファキャッシュ管理
  - 双方向リストの扱いを習得する.
  - OS内部の資源管理法(の初歩)を学ぶ.

- 課題3: shellの作成
  - 文字列処理に慣れる.
  - OSのプロセス管理(の初歩)を学ぶ.

課題とそのねらい (2)

- 課題4 : 疑似DHCPクライアント/サーバの作成
  - UDPを利用したネットワークプログラミング(の初歩)を学ぶ
  - 状態遷移図からプログラムを作成する

- 課題5: 簡易FTPクライアント/サーバの作成
  - TCPを利用したネットワークプログラミング(の初歩)を学ぶ

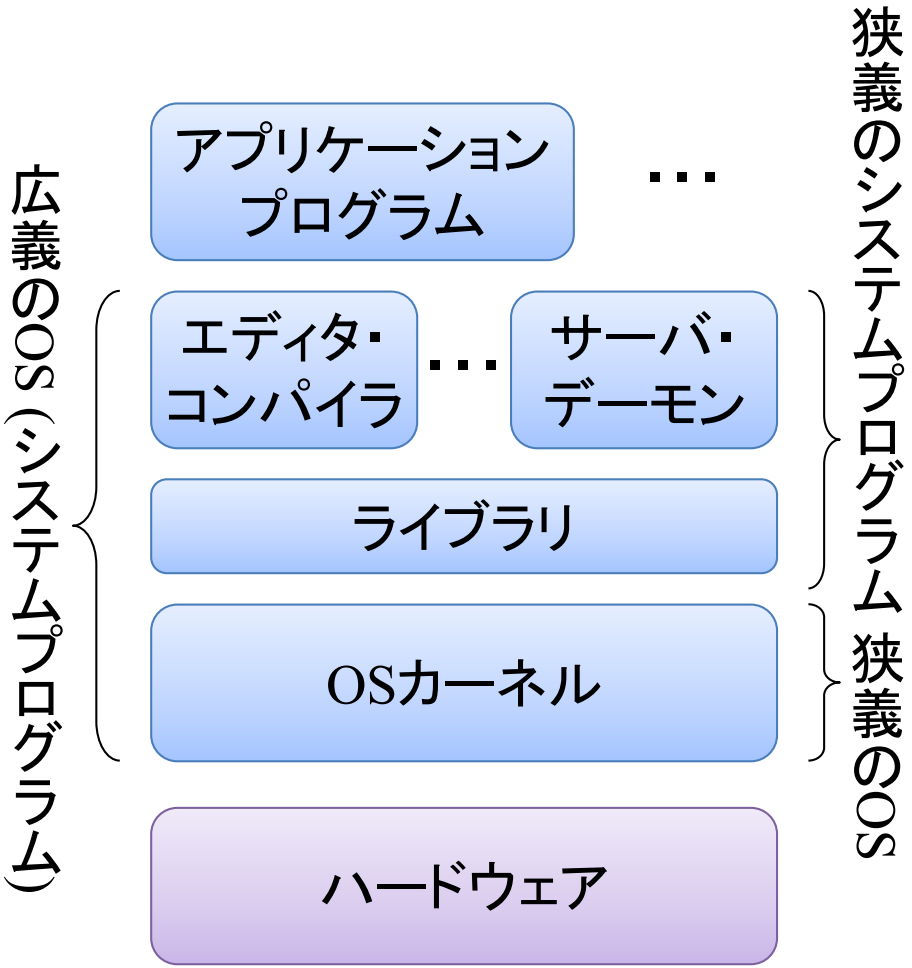
- 課題1を除いて締切までの期間は基本的に4週間
  - 課題1は2週間

課題の採点と成績

- チェックリストに基づきプログラムの動作を確認
  - 正常に動作するか
  - エラー処理が正しいか
  - 注意：ITCのLinuxマシンで動作を確認すること

- 成績は絶対評価
  - S: 90%以上の点数
  - A: 80%以上の点数
  - B: 70%以上の点数
  - C: 60%以上の点数
  - D: 60%未満の点数

システムプログラムとは



- アプリケーションプログラム
  - ユーザが直接利用するプログラム
  - e.g., ゲーム，Webブラウザ
- 狭義のシステムプログラム
  - OSカーネルの上で，アプリケーションプログラムの動作を支えるプログラム
  - e.g., Webサーバ，ライブラリ
- 広義のシステムプログラム
  - OSカーネルも含む
- 厳密な説明は “オペレーティングシステム” を参照のこと

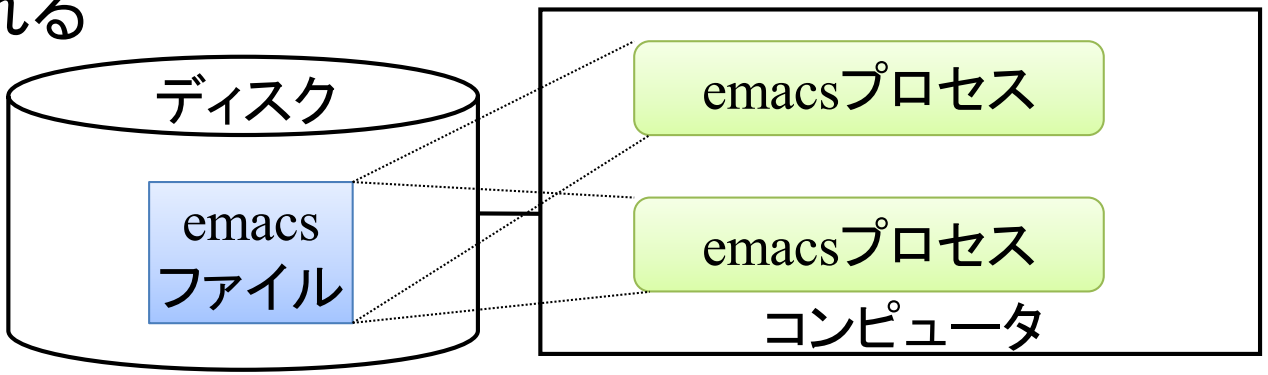
OSの機能

- 資源の管理
  - 資源：CPU，メモリ，ハードディスク，キーボード，画面など．
    - CPU，入出力装置：時間的に分割して使用権を管理
    - メモリ，ディスク領域：空間的に分割して使用権を管理
  - 資源の割当先：プロセス，ユーザ

- 共通したAPIをアプリケーションプログラムに提供
  - API: Application Programming Interface
  - ハードウェアの差異を隠蔽
  - 共通のアプリケーションプログラムが異なったハードウェア上で動作

プログラムとプロセス

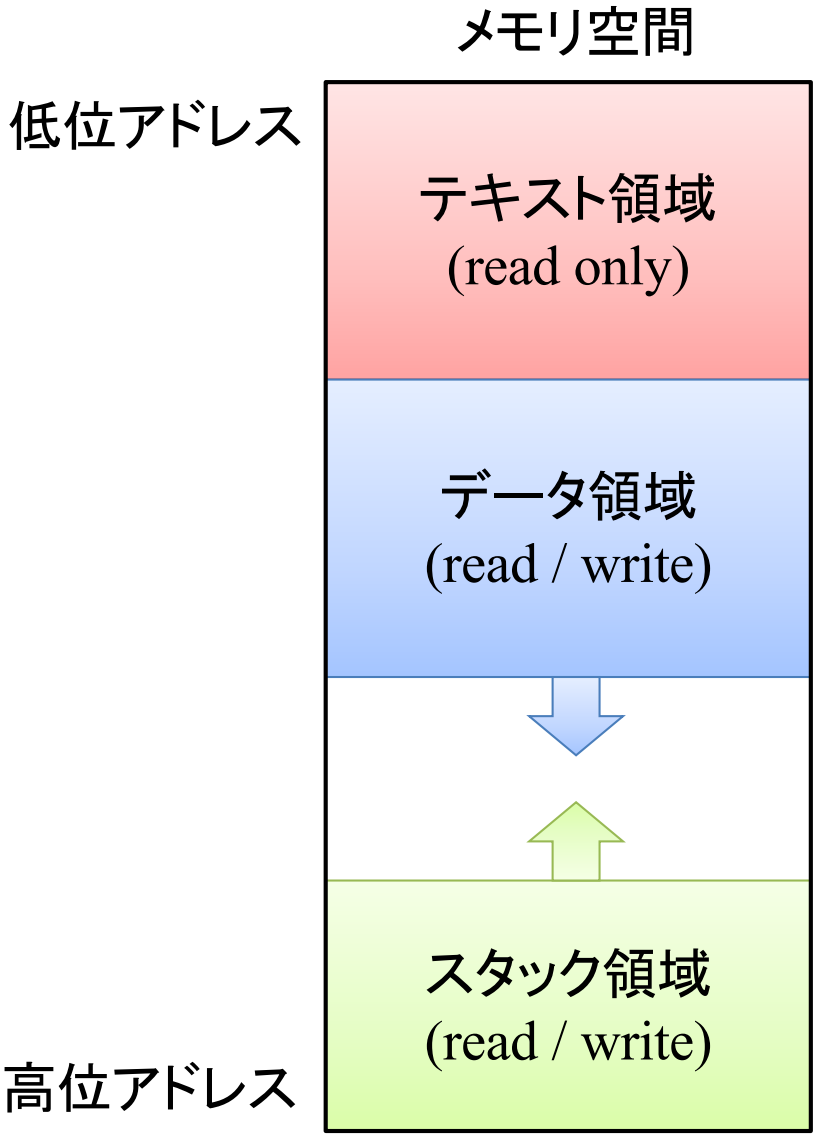
- プログラム
  - コンピュータに対する命令やデータの集まり
  - 通常は “ファイル” としてハードディスクに保存される
- プロセス
  - 実行状態のプログラム
  - CPU時間やメモリ空間などの資源の割り当て対象
  - e.g., “ls” を実行しているプロセス → “lsプロセス” と呼ばれる
  - 同一のプログラムを同時に複数実行すると，その分だけプロセスが生成される



プログラムの実行

- プログラムを実行開始
  - e.g., “emacs” と入力
- OS内部で新しいプロセスが生成される
  - 6回目の講義参照 (5/18の予定)
- ディスクからプログラムファイルがメモリにロードされる
  - e.g., /usr/local/bin/emacs というファイルがロードされる
  - テキスト領域，データ領域，スタック領域(後述)が作られる
- プロセスがemacsプログラムを実行開始 → emacsプロセス

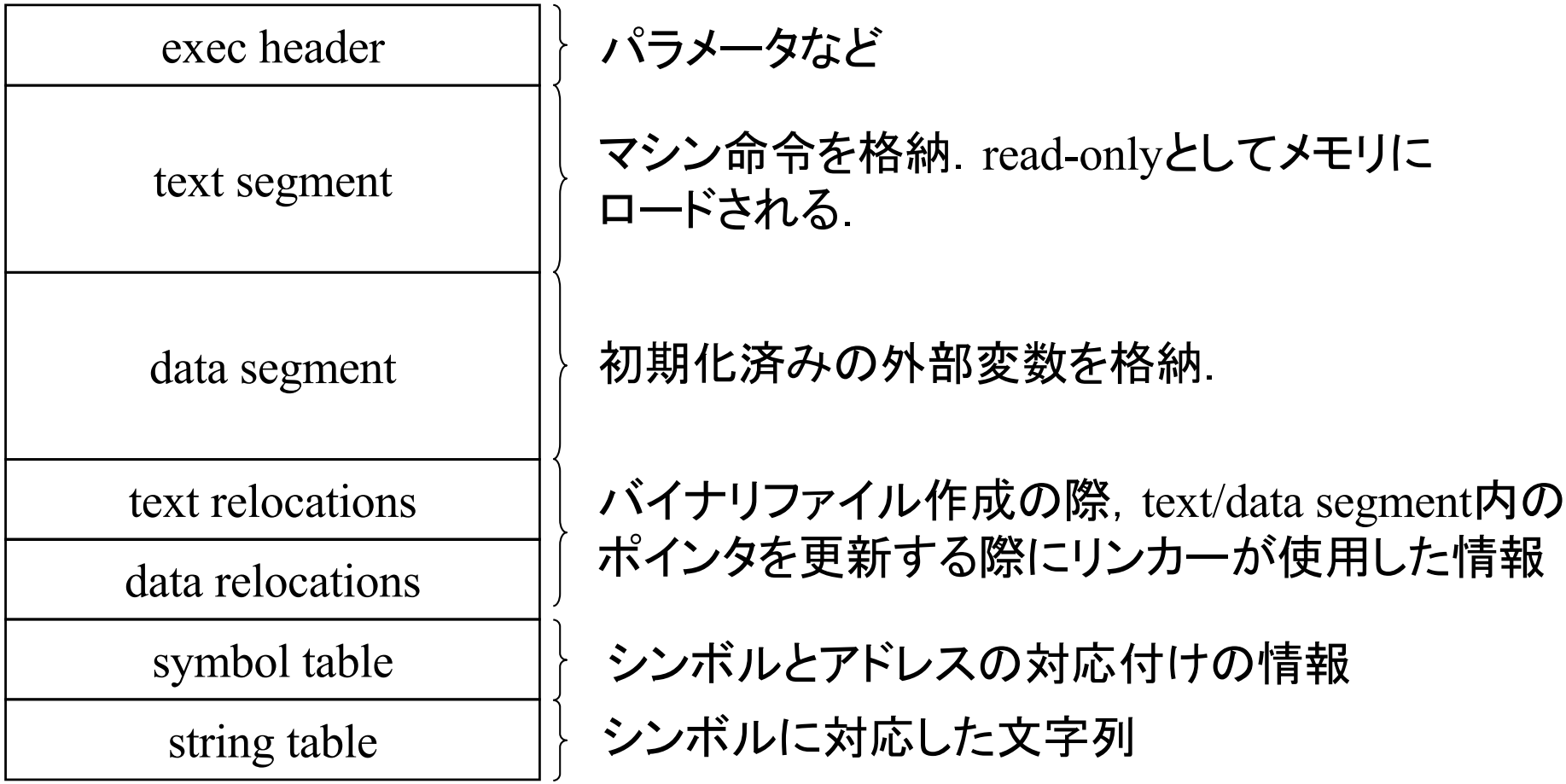
メモリ上のイメージ



- テキスト領域
  - 命令が格納される
  - read only
- データ領域
  - 外部変数が格納される
  - read / write 可能
  - malloc( ) で拡張される
- スタック領域
  - 局所変数や関数の戻り番地が格納される
  - 関数呼び出しで拡張，リターンで縮小

実行ファイル(a.out)のフォーマット

- a.out形式：昔のUNIXの実行ファイルの形式



実行ファイル形式: ELF

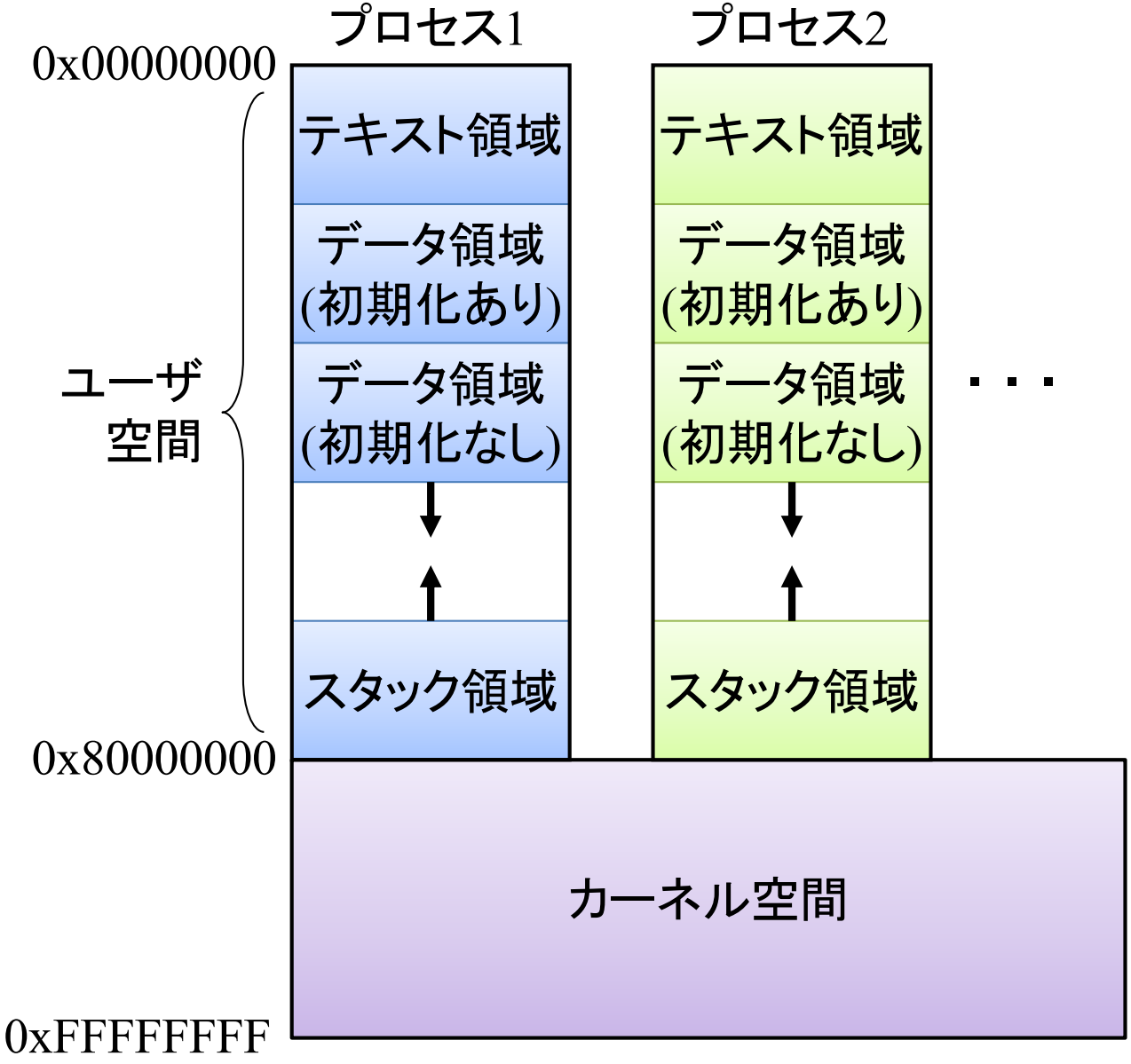
- ELF: Executable and Linking Format
- Linuxの標準的な実行ファイル形式
- ファイル形式の判定

```
% file a.out
a.out: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 ...
%
```

- ELF形式のヘッダ情報の表示

```
% readelf -h a.out
ELF ヘッダ:
  マジック: ....
...
%
```

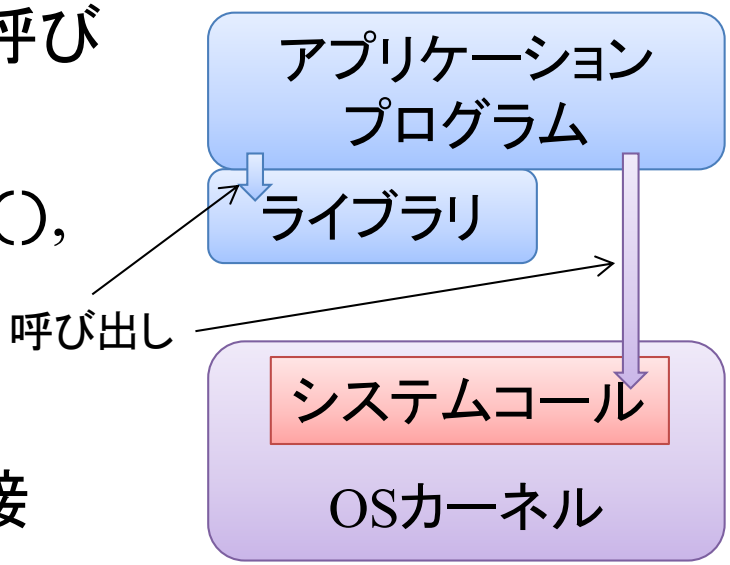
仮想メモリ空間 (32ビットの場合)

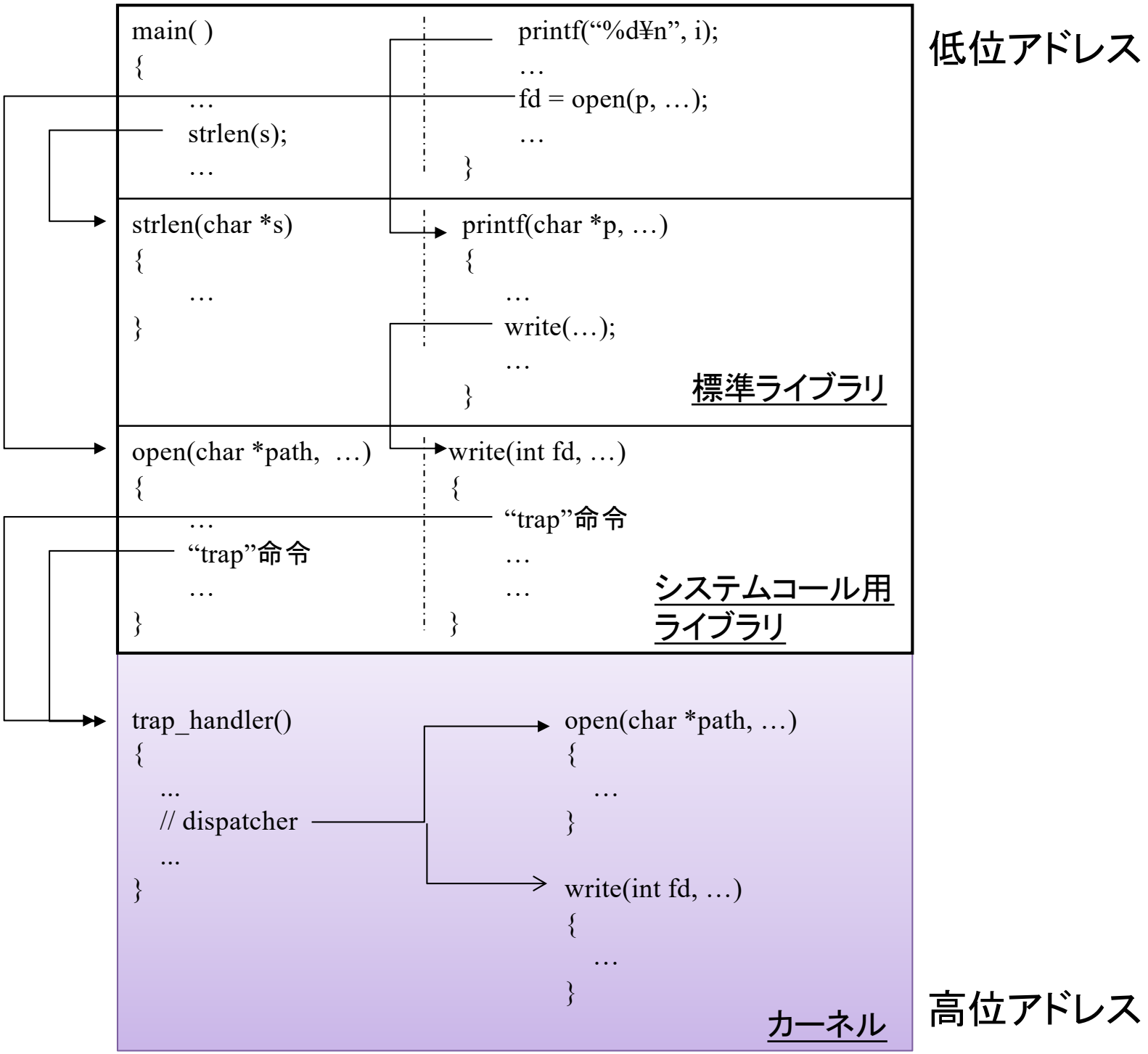


- カーネル空間はすべてのプロセスによって共有される
- 各プロセスは32ビットの仮想アドレス空間をもつ (0x00000000～0xFFFFFFFF)
- 通常、下位番地の空間はユーザ空間となる

ライブラリとシステムコール

- ライブラリ
  - よく使われる関数をまとめたもの
  - 通常プログラムにリンクされる
  - ライブラリの中でシステムコールを呼び出しているものもある
  - e.g., `strlen()`, `printf()`, `scanf()`, etc.
- システムコール
  - カーネルが提供するサービスを直接呼び出すこと
  - e.g., `open()`, `read()`, `write()`, `close()`, etc.




main( )
{
…
strlen(s);
…
printf(“%d¥n”, i);
…
fd = open(p, …);
…
}
低位アドレス
strlen(char *s)
{
…
}
printf(char *p, …)
{
…
write(…);
…
}
標準ライブラリ
open(char *path, …)
{
…
“trap”命令
…
}
write(int fd, …)
{
“trap”命令
…
…
}
システムコール用
ライブラリ
trap_handler()
{
...
// dispatcher
...
}
open(char *path, …)
{
…
}
write(int fd, …)
{
…
}
カーネル
高位アドレス

manコマンドを活用しよう！

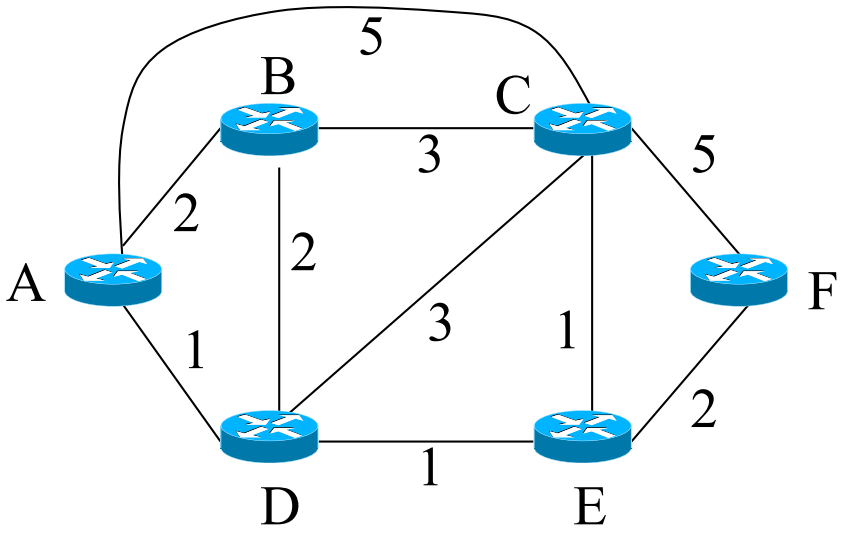
- syntax: man [section] name …

- コマンド，システムコール，ライブラリなどのマニュアル
  - 同じ名前が複数のセクションにあるときには、sectionを指定
    e.g., “man 2 write” : システムコールのwriteのマニュアル

- sectionの種類
  - 1: コマンド
  - 2: システムコール
  - 3: ライブラリ関数
  - 4: カーネルインタフェース
  - 5. ファイルフォーマット
  - 6. ゲーム
  - 7. misc.
  - 8. システム管理者用

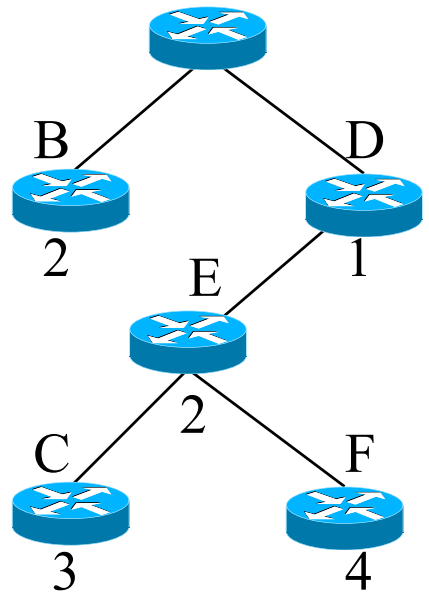
課題1：Dijkstra's Shortest Path First Algorithm (1)

- 経路計算方法の一種
  - 前提：すべてのノードがトポロジを知っている
    - トポロジ：ノード間の接続状態
  - 問題：各ノードを根とした最小コストの木構造(shortest path tree)を計算



数字はリンクのコスト

例：Aを根とした shortest path treeを計算



各ノードで計算

課題1 : Dijkstra's Shortest Path First Algorithm (2)

- 記法
  - N : ノードの集合 (入力)
  - u : 始点ノード
  - $c(i, j)$: ノードi からノードj へのリンクコスト. i と j が隣接していない場合は無限大 (入力).
  - $D(v)$: 始点から終点 v までの反復計算における現在の最小コストの値 (出力).
  - $p(v)$: 始点から終点 v への現在の最小コスト経路上における v の1つ手前のノード (出力).
  - N' : 最小コスト経路が最終的にわかっているノードの集合. vまでの最小コスト経路が分かっていれば, vはN'の要素.

課題1：Dijkstra's Shortest Path First Algorithm (3)

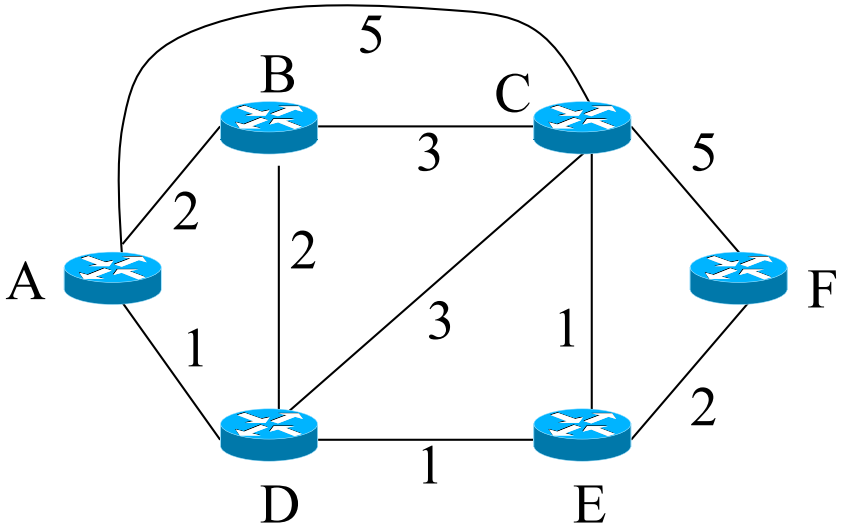
ノード*u* から他のすべてのノード*v* への経路の計算

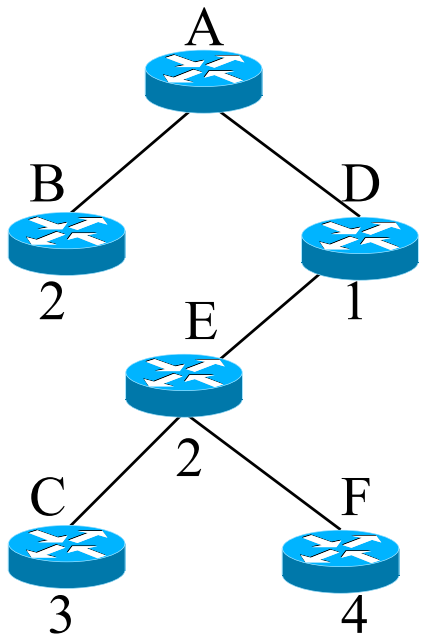
```
1   Initialization:
2      N' = {u}
3      for all nodes v
4         if v is a neighbor of u
5            then D(v) = c(u, v), p(v) = u
6         else D(v) = ∞
7    Loop
8       find w not in N' such that D(w) is a minimum, and add w to N'
9       update D(v) (and p(v))for each neighbor v of w and not in N':
10         if D(v) > D(w) + c(w, v)
11            then D(v) = D(w) + c(w, v), p(v) = w
12   until N' = N
```

課題1：Dijkstra's Shortest Path First Algorithm (4)

ノードAでの実行例

Step	N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD	2,A	4,D		2,D	∞
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					





ノードAの転送表

dst	next
B	direct
D	direct
C	D
E	D
F	D

課題1：Dijkstra's Shortest Path First Algorithm (5)

- 課題：前ページのネットワークにおいて，各ノードの$D(v)$および$p(v)$を求める．
  (v: A, B, …, F)
  - A=0, B=1, …, F=5 とする
- 入力

```
#define NNODE 6
#define INF      100  // infinity
int cost[NNODE][NNODE] = {
    {   0,   2,   5,    1,  INF,  INF},
    {   2,   0,   3,    2,  INF,  INF},
    {   5,   3,   0,    3,    1,    5},
    {   1,   2,   3,    0,    1,  INF},
    {INF,  INF,  1,    1,    0,    2},
    {INF,  INF,  5,  INF,    2,    0}
};
```

- 出力

```
int dist[NNODE];
int prev[NNODE];
```

- 出力例

```
root node A:
      [A,A,0] [B,A,2] [C,E,3] …
root node B:
      …
…
root node F:
      …
```

直前ノード　終点ノード　コスト

- 指定出力形式に従うこと
  - 異なる形式の場合減点対象

課題1 : Dijkstra's Shortest Path First Algorithm (6)

- 提出方法
  - ITCのLinux環境で動作を確認すること.
  - 1つのソースファイルにまとめる. ファイル名は任意.
  - ソースファイルの先頭にコメントとして学籍番号と氏名を記入.
    - 例 : // 12345678 寺岡文男
    - 未記入の場合は減点します
  - CANVASの「課題」に提出
  - 締切 : 2022年4月25日(月) 20:00(JST)

- 他人のプログラムをコピーしないように！
  - コピーと思われるものはすべて0点にします.