システムプログラム
第8回

創域理工学部 情報計算科学科
松澤　智史

本日の内容

- システムプログラムの一部であるOSについて学ぶ
  - 第8回では，OSの機能の一つである「ネットワーク管理」を学ぶ
    - 入出力(ファイルディスクリプタ)について学ぶ

OSのネットワーク管理

メモリを共有しない対象とのデータ共有

- 他のプロセスとの通信
- 外部端末との通信

プロセス間通信

Inter Process Communication(IPC)とも呼ばれる

- 共有メモリ
  - メモリ空間を共有する，セマフォなどの同期，シグナルなどの機構を使用する
- PIPE
  - あるプロセスの出力を別のプロセスの入力につなげる
- UNIXドメインソケット
  - 同一ホスト内でのアプリケーション間の通信をサポートする
- インターネットドメインソケット
  - インターネットプロトコルで接続されたアプリケーション間の通信をサポートする

外部端末との通信

- 通信プロトコルを解釈，実行するソフトウェア
  - TCP/IPプロトコルスタック
  - AppleTalkプロトコルスタック
  - SMB/CIFSプロトコルスタック
    etc
- 通信デバイスの管理
  - Bluetooth
  - IEEE802.11
  - Ethernet

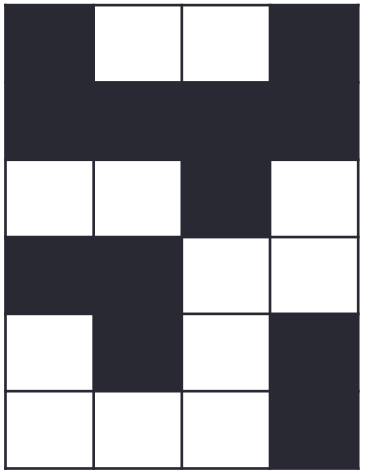
プロトコルとスタック

- プロトコル
  - 通信規約
  - データの解釈の仕方
  - 双方で事前に決まり事を共有

- スタック
  - 役割ごとにソフトウェアをわける(階層構造)
  - 各階層のソフトウェアの出力を次の階層のソフトウェアの入力へ渡す

プロトコル

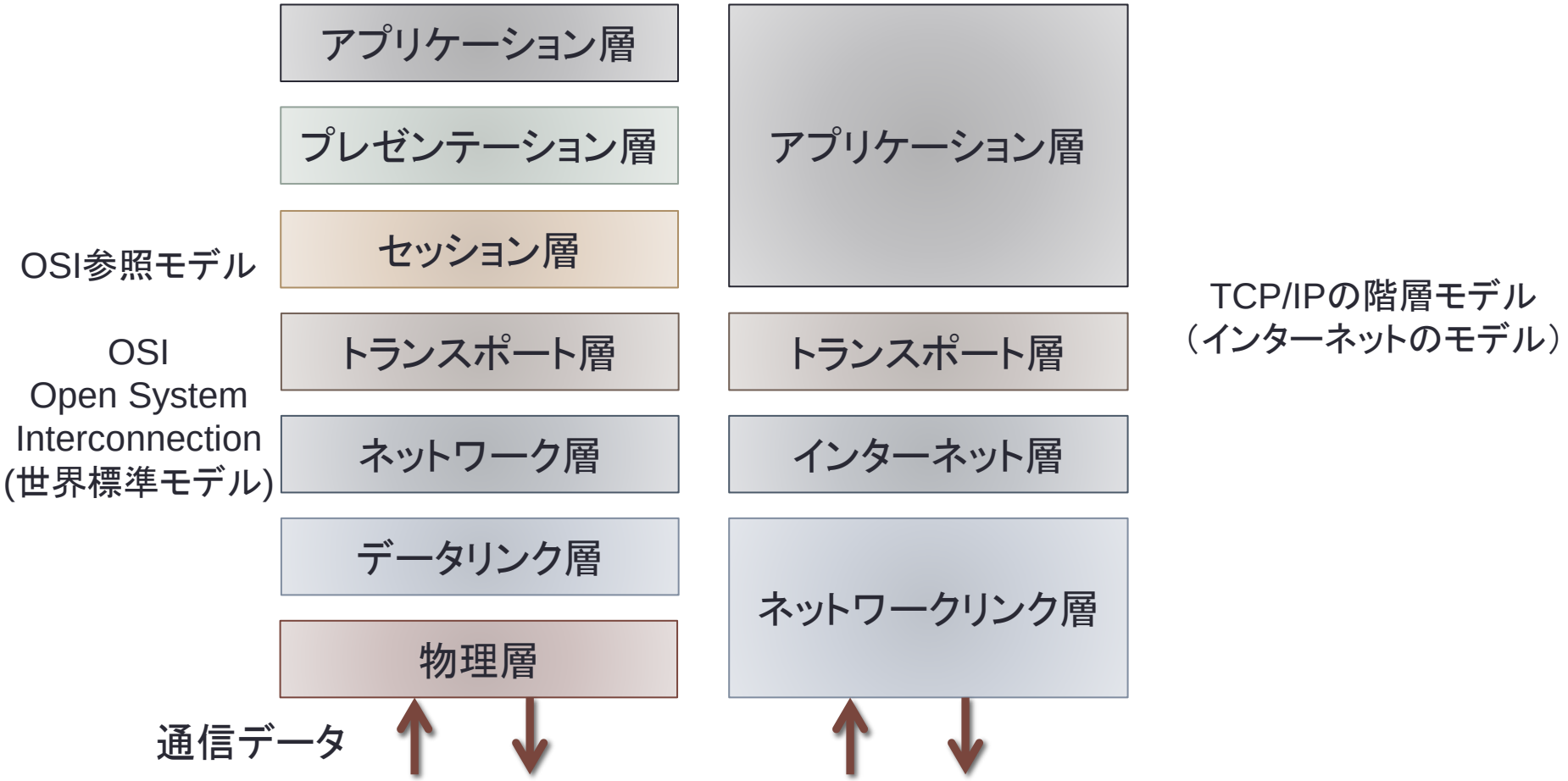
- 例　2色画像の情報送信
  - データ定義
    - 黒→0
    - 白→1
  - 通信方法の定義
    - 縦方向の座標(3bit) 横方向の座標(3bit)白or黒(1bit)
    - 計7ビットで1ピクセルの情報を定義
  - 通信例
    - 0000001　→x=0,y=0の位置は黒
    - 0010000　→x=1,y=0の位置は白



スタック

- 相手を識別してデータを届ける役割のソフトウェア
- 信頼性を確保するためのソフトウェア
- 特定アプリケーションのメモリに受け渡す役割のソフトウェア

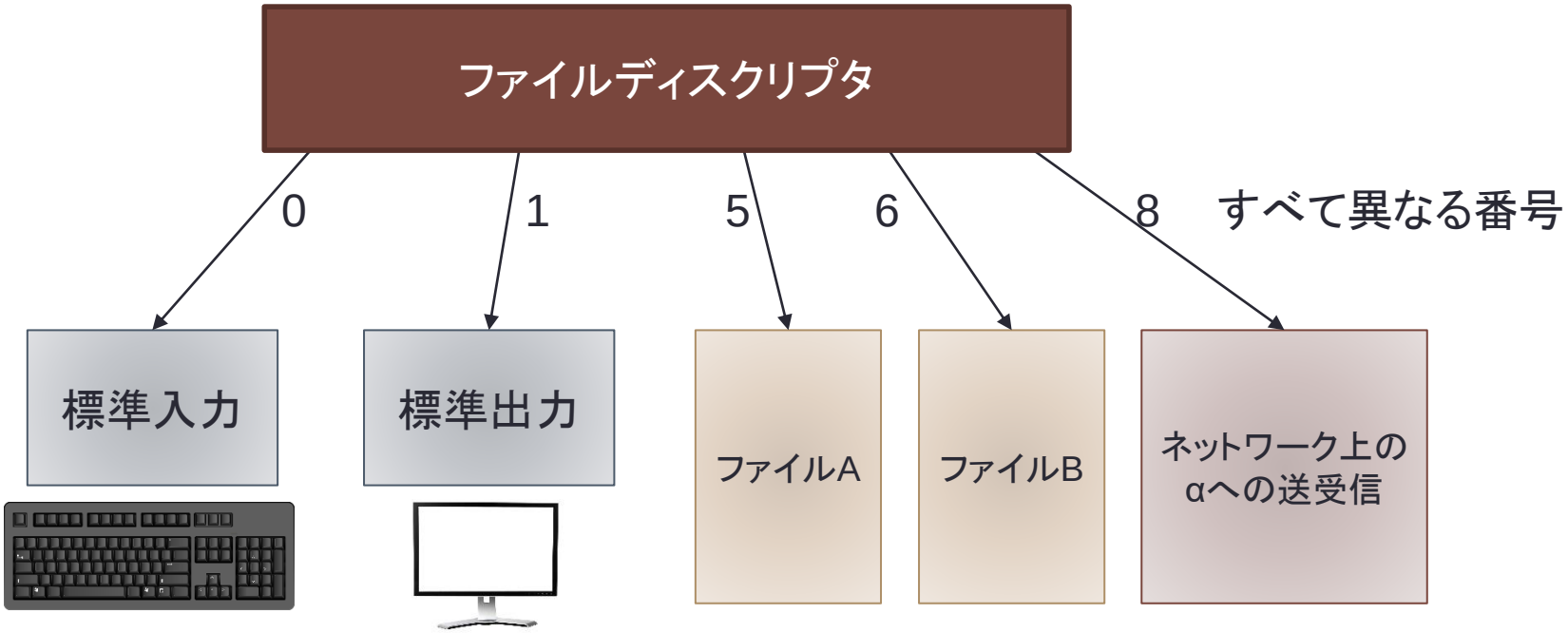
ネットワークの階層構造



OSでサポートすべき階層

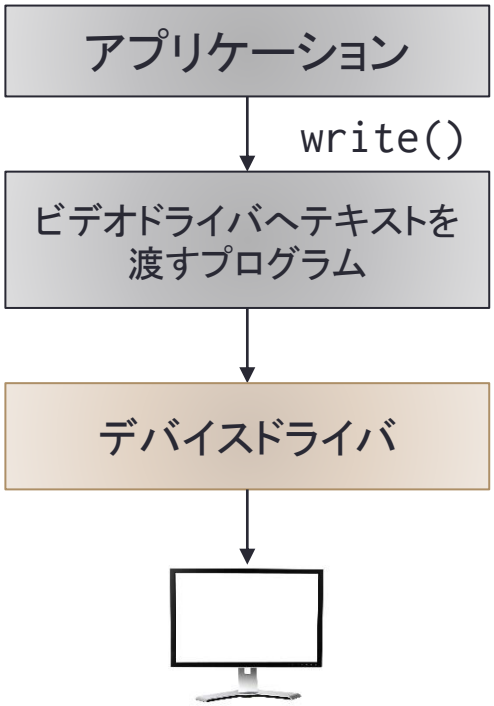
OSのファイルディスクリプタ

- OSは入出力のストリームをファイルディスクリプタ(番号)で管理
- ファイルディスクリプタは必ず一意(他と同じ番号ではない)で割り当て
- 標準入力(0)，標準出力(1)，標準エラー出力(2)はあらかじめ動作
- プロセスごとにファイルディスクリプタは独立



標準出力への書き込み

```
File  Edit  Options  Buffers  Tools  C  Help
#include <stdio.h>
#include <unistd.h>

int main(){
  write(1,"aaa\n",4);

  printf("aaa\n");
}

-:---   stdo.c          All (1,18)
```



printf()はwirte()を呼び出す関数

write()の第一引数がファイルディスクリプタ番号

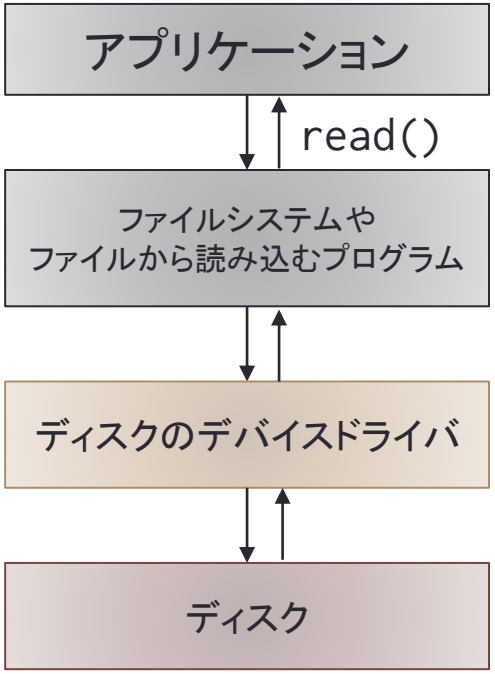
ファイルからの読み込み

```
File  Edit  Options  Buffers  Tools  C  Help
#include <stdio.h>

int main() {
  FILE *fp;
  int f;
  char buf[1024];

  fp = fopen("ftoken.c","r");

  while(1) {
    f = fscanf(fp,"%s",buf);
    if (f == EOF) break;
    printf("%s\n",buf);
  }

  fclose(fp);
}
U:---   ftoken.c        All (1,18)      (C
```

情報科学演習2で行った
ファイルからの読み込みサンプルプログラム

```
File  Edit  Options  Buffers  Tools  C  Help
#include <stdio.h>

int main() {
  FILE *fp;
  int f;

  fp = fopen("ftoken.c","r");
  printf("%d\n",fileno(fp));
  fclose(fp);
}
-:---   ftoken.c        All (11,0)      (C/*l
```

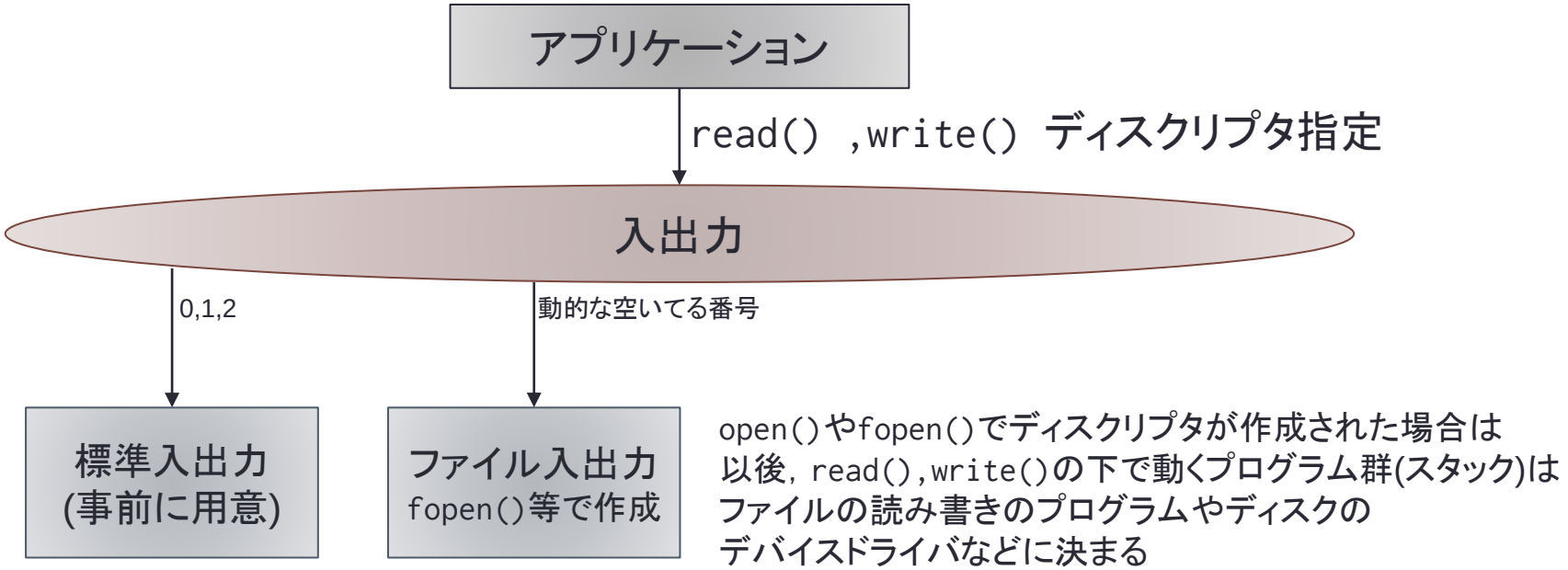
ファイルディスクリプタ番号が表示される



FILE構造体(OSによって中身は違う)にはディスクリプタの番号がメンバ変数として格納されている
FILE構造体からディスクリプタ番号を取得する関数は`fileno(FILE *fp)` (この関数はどのOSでも実装済)

各デバイスへの入出力

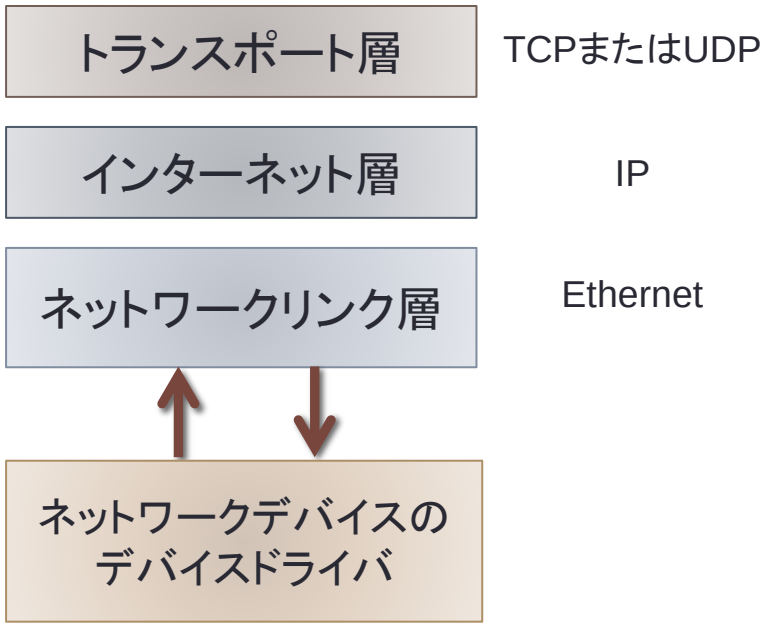
- OSはデバイスに依存しないread(),write()の関数を提供
- read(),write()にはファイルディスクリプタを指定する
- ディスクリプタの割り当て時に使うプログラムのスタック群を指定する



socket() システムコール

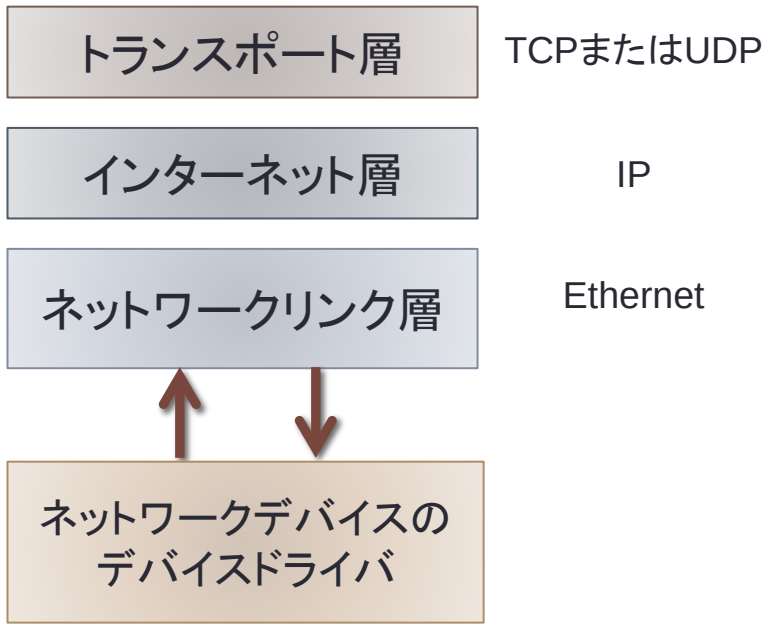
- ネットワークのファイルディスクリプタ作成

- 引数によって使うプロトコルスタック(プログラムスタック)を選択可能
  - TCP/IP/Ethernet
  - UDP/IP/Ethernet
  - IP/Ethernet
  - Ethernet



補足:各層の役割

- トランスポート層
  - アプリケーションから渡されたデータに信頼性の有(TCP)無(UDP)の処理を施す
- インターネット層
  - データを遠隔ノードに到達させるための情報を付与する
  - 付与した情報を元にデータ配送される
- リンク層(データリンク層)
  - 隣接ノードに渡すための情報を付与する
- デバイスドライバ
  - ハードウェア仕様に従った動作に変換する



socket() システムコール

- 書式
  - `int socket(int domain , int type , int protocol );`

- 三つの引数で，プロトコルスタックを決定
  - `domain`: プロトコルスタックを指定 (IPv4or6,ローカル通信，AppleTalkなどを指定)
  - `type`: 通信方式を指定(信頼性のあるなし，生データなどを指定)
  - `protocol`: ソケットによって使用される固有のプロトコルを指定(指定なしは0)

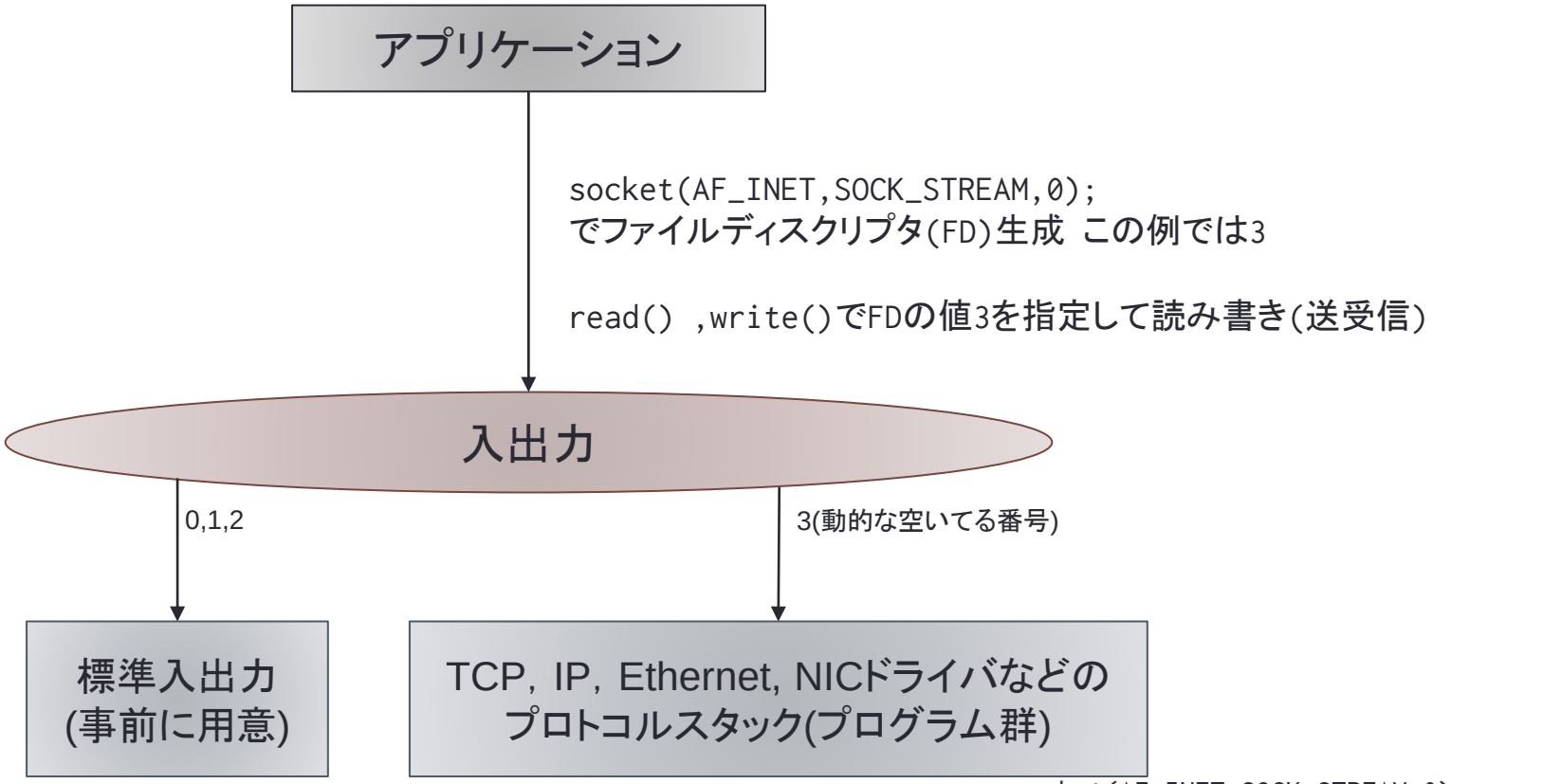
  (TCP/IP以外の通信もサポートするため，具体的なTCPなどの名前でなく，あいまいな点に注意)

- 返り値の`int`はファイルディスクリプタ番号

socket() システムコール 使用例

- TCP/IP
  - `socket(AF_INET, SOCK_STREAM, 0);`
- UDP/IP
  - `socket(AF_INET, SOCK_DGRAM, 0);`
- IP
  - `socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);`
- 生(RAW)データ
  - `socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL)))`

TCP/IPの通信


アプリケーション
socket(AF_INET,SOCK_STREAM,0);
でファイルディスクリプタ(FD)生成 この例では3
read() ,write()でFDの値3を指定して読み書き(送受信)
入出力
0,1,2
3(動的な空いてる番号)
標準入出力
(事前に用意)
TCP, IP, Ethernet, NICドライバなどの
プロトコルスタック(プログラム群)
socket(AF_INET,SOCK_STREAM,0);
によりOSやカーネルで用意してある
TCP,IP,ethernetのプログラムが選ばれる

TCP/IPの通信プログラム

File Edit Options Buffers Tools C Help

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h> // memset()
#include <arpa/inet.h> // inet_addr()
#include <unistd.h> //close write

int main() {
  struct sockaddr_in server;
  int sock;
  char buf[32];
  int n;

  /* ソケットの作成 */
  sock = socket(AF_INET, SOCK_STREAM, 0);

  /* 接続先指定用構造体の準備 */
  server.sin_family = AF_INET;
  server.sin_port = htons(12345);
  server.sin_addr.s_addr = inet_addr("127.0.0.1");

  /* サーバに接続 */
  connect(sock, (struct sockaddr *)&server, sizeof(server));

  /* サーバからデータを受信 */
  memset(buf, 0, sizeof(buf));
  n = read(sock, buf, sizeof(buf));

  printf("%d, %s\n", n, buf);

  /* socketの終了 */
  close(sock);

  return 0;
}
```

U:--- tcpclient.c All (36,1) (C/*l Abbrev) 8:39午後 0.97

クライアントプログラム

File Edit Options Buffers Tools C Help

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h> //close() write()

int main(){
  int sock0;
  struct sockaddr_in addr;
  struct sockaddr_in client;
  int len;
  int sock;

  sock0 = socket(AF_INET, SOCK_STREAM, 0);
  addr.sin_family = AF_INET;
  addr.sin_port = htons(12345);
  addr.sin_addr.s_addr = INADDR_ANY;

  bind(sock0, (struct sockaddr *)&addr, sizeof(addr));
  listen(sock0, 5);

  while (1) {
    len = sizeof(client);
    sock = accept(sock0, (struct sockaddr *)&client, &len);
    write(sock, "HELLO", 5);

    close(sock);
  }
  close(sock0);
  return 0;
}
```

U:--- tcpserver.c All (1,18) (C/*l Abbrev) 8:42午後 0.

サーバプログラム

ターミナル2つ起動し，サーバ，クライアントの順で実行する
サーバより送られるHELLOの文字がクライアント側で表示される

AF_INETは IPv4
SOCK_STREAMは信頼性あり
※この時点でTCP/IPに決まるため第三引数は0(指定なし)

関数補足

File Edit Options Buffers Tools C Help

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h> // memset()
#include <arpa/inet.h> // inet_addr()
#include <unistd.h> //close write

int main() {
  struct sockaddr_in server;
  int sock;
  char buf[32];
  int n;

  /* ソケットの作成 */
  sock = socket(AF_INET, SOCK_STREAM, 0);

  /* 接続先指定用構造体の準備 */
  server.sin_family = AF_INET;
  server.sin_port = htons(12345);
  server.sin_addr.s_addr = inet_addr("127.0.0.1");

  /* サーバに接続 */
  connect(sock, (struct sockaddr *)&server, sizeof(server));

  /* サーバからデータを受信 */
  memset(buf, 0, sizeof(buf));
  n = read(sock, buf, sizeof(buf));

  printf("%d, %s\n", n, buf);

  /* socketの終了 */
  close(sock);

  return 0;
}
```

U:--- tcpclient.c All (36,1) (C/*l Abbrev) 8:39午後 0.97

- クライアント(接続を能動的に確立するノード)
  - connect()
    - 指定された構造体に格納されている相手ノードへ
      TCPコネクションを確立しようと試みる
- サーバ(接続を受動的に受け入れるノード)
  - bind()
    - 指定された構造体に格納されている状態へ遷移
  - listen()
    - 受付開始　同時接続する等を指定
  - accept()
    - 実際に接続要求が来るまで待機
    - 接続要求が来た場合は新しいファイルディスクリプタを生成して
      以後そのノードとの送受信をそのディスクリプタに任せる

関数補足

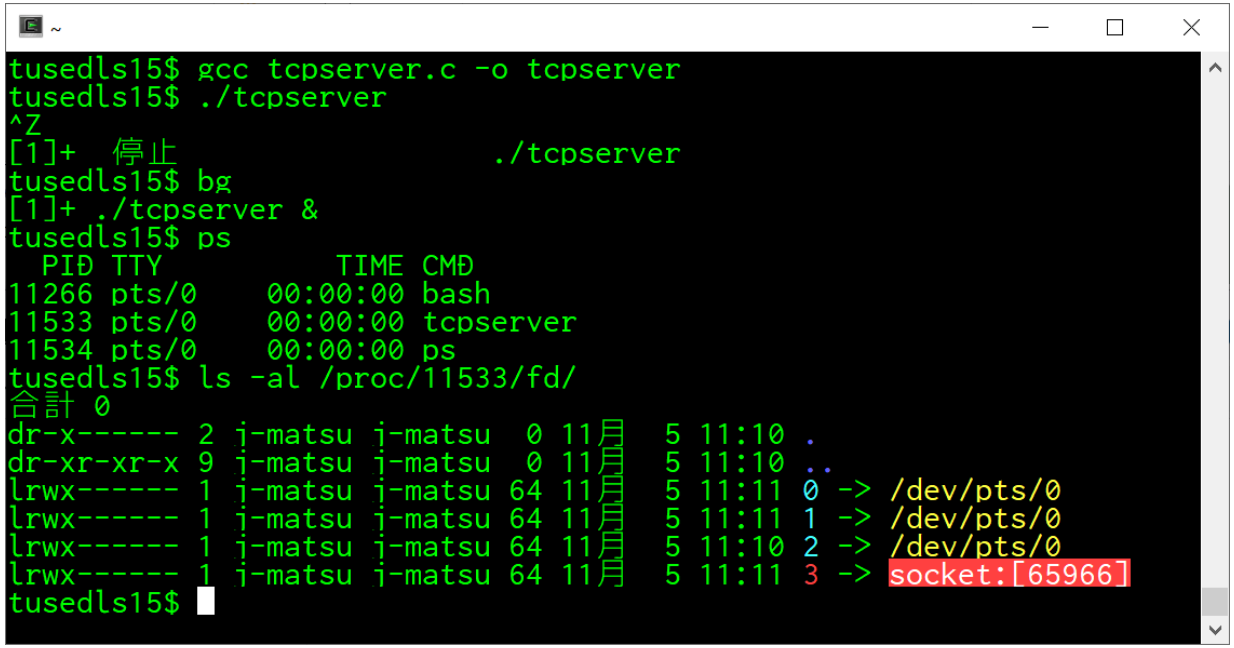
File Edit Options Buffers Tools C Help

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h> //close() write()

int main(){
  int sock0;
  struct sockaddr_in addr;
  struct sockaddr_in client;
  int len;
  int sock;

  sock0 = socket(AF_INET, SOCK_STREAM, 0);
  addr.sin_family = AF_INET;
  addr.sin_port = htons(12345);
  addr.sin_addr.s_addr = INADDR_ANY;

  bind(sock0, (struct sockaddr *)&addr, sizeof(addr));
  listen(sock0, 5);

  while (1) {
    len = sizeof(client);
    sock = accept(sock0, (struct sockaddr *)&client, &len);
    write(sock, "HELLO", 5);

    close(sock);
  }
  close(sock0);
  return 0;
}
```

U:--- tcpserver.c All (1,18) (C/*l Abbrev) 8:42午後 0.

- クライアント(接続を能動的に確立するノード)
  - connect()
    - 指定された構造体に格納されている相手ノードへ
      TCPコネクションを確立しようと試みる
- サーバ(接続を受動的に受け入れるノード)
  - bind()
    - 指定された構造体に格納されている状態へ遷移
  - listen()
    - 受付開始　同時接続する等を指定
  - accept()
    - 実際に接続要求が来るまで待機
    - 接続要求が来た場合は新しいファイルディスクリプタを生成して
      以後そのノードとの送受信をそのディスクリプタに任せる

プロセスのファイルディスクリプタ確認



/proc/プロセスID/fd/
にプロセスが現在使用しているファイルディスクリプタが記載されている

現在は3がsocket()システムコールにより生成されている

※プロセスの強制終了は
kill -9 プロセスID
で行う

まとめ

- ファイルディスクリプタ
  - 各入出力I/Oは，ファイルディスクリプタの番号で識別され，アプリケーションからは
    その識別番号へのwrite(),read()で読み書きできる
  - ネットワークのI/Oは，socket()システムコールでファイルディスクリプタが生成される
- socket()システムコール
  - あらゆるプロトコルスタックを使用するためのディスクリプタ生成関数(システムコール)
  - TCP/IPなどのインターネットを前提としたプロトコルもサポートする

質問あればどうぞ