今回は、プログラムで他のプログラムを起動する方法を学ぶ。

1. Fork-Exec 方式

   UNIX で新しくプログラム起動するために用いられてきた方式のことである。fork(プロセスの生成) と exec(プログラムロード・実行) の２段階で目的を達成する。

   (a) fork

   `fork` システムコールを用い、新しいプロセスを作成する。

   書式： `#include <unistd.h>`
   `int fork(void);`

   解説： プロセスのコピーを作る。親プロセスには子プロセスの `PID` が返される。
   エラー時はが返される。

   `fork` システムコールを実行した「プロセスがコピーされ」る。元のプロセスを**親プロセス**、コピーして作ったプロセスを**子プロセス**と呼ぶ。図 1 に fork の様子を、リスト 1 に `fork` システムコールを実行するプログラムの例を、以下に fork の処理手順を示す。

   i. 親プロセスがプログラム実行中に `fork` システムコールを実行する。
   ii. 新しいプロセス（子プロセス）が作られ、親プロセスの内容がコピーされる。
   iii. 子プロセスには、メモリ空間（プログラム、変数（データ）、スタック）、プロセスの状態（どのファイルをオープン中か、シグナルハンドラの登録等）、CPU の状態（CPU レジスタの値、SP の値、PC の値、フラグの値）等、全ての情報がコピーされる。ただし、プロセス番号 (PID:Process ID) 値は親子プロセスで異なる。
   iv. 親プロセスは `fork` システムコールを終了しプログラムの実行を再開する。この時、`fork` システムコールは子プロセスの PID を返す。
   v. 子プロセスは `fork` システムコールを呼出した瞬間のコピーなので、`fork` システムコールが終了するところからプログラム実行を開始する。この時、`fork` システムコールは `0` を返す。

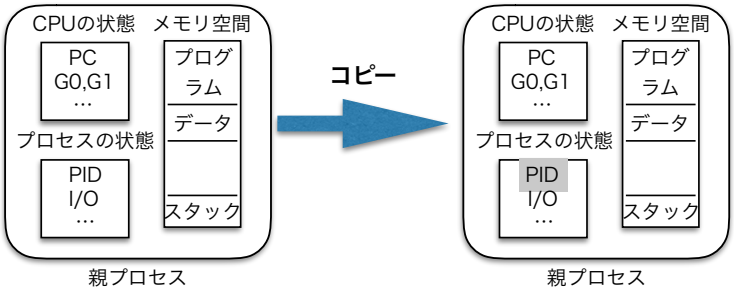

図 1: fork の仕組み

リスト 1: fork の使用例

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main() {
```

```
5   int x = 10;
6   int pid;
7
8   pid = fork();                                   // この瞬間にプロセスがコピーされる
9   if (pid<0) {
10    fprintf(stderr, "fork でエラー発生\n");  // エラーの場合
11    return 1;
12  } else if (pid!=0)   {                          // 親プロセスだけが以下を実行する
13    x = 20;                                       //   親プロセスの x を書き換える
14    printf("親␣pid=%d␣x=%d\n", pid, x);
15  } else {                                        // 子プロセスだけが以下を実行する
16    printf("子␣pid=%d␣x=%d\n", pid, x);      //   子プロセスの x は初期値のまま
17  }
18  return 0;
19 }
20
21 /*  実行例
22 $ forktest
23 親 pid=8079 x=20              // 親プロセスの出力
24 子 pid=0 x=10                 // 子プロセスの出力(x の値に注目)
25 */
```