# 情報処理システム Ⅱ ・第 8 回

2025 年 5 月 30 日

配布資料の内容



## 8.1 ライブラリ

OS を (カーネルだけではなく) 広く捉えた場合、OS がアプリケーションプログラムなどのユーザプログラムに提供するものの一つに**ライブラリ**があります。ライブラリは、汎用性の高い機械語プログラムのサブルーチン群[1]を集めたものです。たとえば、C 言語でよく使われる `printf` や `exit` などの関数は**標準 C ライブラリ**と呼ばれるライブラリの一部であり、それぞれの機械語プログラムは、一種のプログラムファイルに格納されており、Linux や Windows などの OS の一部となっています。

また、特権命令を直接実行することのできないユーザプログラムが、計算機システムの各種の資源にアクセスしたい場合には、ソフトウェア割り込みを発生される機械語命令によって、システムコールを行い、カーネルに必要な処理を行ってもらいますが、多くの OS では、C 言語などの高級言語で書かれたプログラムから容易にシステムコールを行うことができるように、システムコールを行うサブルーチン群 (C 言語の場合は関数群) を納めたライブラリを提供しています。

このようなライブラリによって、ユーザプログラムは、システムコールを含めた様々な仕事を容易に行うことができるようになります。

メモ

## 8.2 オブジェクトファイルのリンク

通常、C 言語などの高級言語で書かれたプログラムを、コンパイラで機械語プログラムに変換しても、単体で機能できるような完全な機械語プログラムとはなりません。

myprog.c

```
#include <stdio.h>

int main()
{
    int n;

    printf("n = ");
    scanf("%d", &n);
```

[1] C 言語における関数

```
    printf("%d*%d = %d\n", n, n, n*n);
    return 0;
}
```

たとえば、上のような C 言語のプログラムをコンパイルすると、まず、このソースプログラムを機械語に変換した結果を納めたオブジェクトファイルが作成されます。Linux 環境では、このオブジェクトファイルの名前は、通常、myprog.o となりますが[2]、この myprog.o に含まれている機械語プログラムは、関数 main の定義の C プログラムに相当する部分だけであり、たとえば、関数 printf や scanf の働きをする機械語プログラムは含まれていません。また、プロセスが起動した後に必要な処理を行って関数 main を呼び出し、main から戻ってきた (return してきた) 際の後処理を行って、自分のプロセスを終了させるシステムコールを行う機械語プログラムも含まれていません。このため、myprog.o だけでは、このプログラムを新たなプロセスとして起動することはできません。関数 main を呼び出す (後者の) 機械語プログラムは、C ランタイムオブジェクトと呼ばれます。

cc -o myprog myprog.c というコマンドを実行すると、コンパイラ cc は myprog.c をコンパイルした結果を myprog.o というオブジェクトフィルに保存し、その後、この myprog.o と C ランタイムオブジェクト (ファイル) を結合してプログラムファイル myprog を作成します[3]。この結合を行う作業を**リンク (link)** と呼びます[4]。

メモ

**動的リンク**　ここまでで得られたプログラムファイル myprog には、まだ、関数 printf や scanf の機械語プログラムは含まれていません。これらの関数は、C 言語の標準ライブラリに含まれており、それぞれの機械語プログラムは一つのファイルの中に記録されています。myprog が起動されると、そのプログラムファイル中に指定されている特定のオブジェクトファイルが、カーネルによって、myprog のプロセスの仮想アドレス空間の一部に取り込まれ[5]、その機械語プログラムが実行されます。このプログラムは**動的リンカ**と呼ばれ、myprog のプログラムファイル中に記録されている情報に基づき、myprog が使用しているライブラリファイルの内容を、mmap システムコールを

[2]Windows 環境では myprog.obj となります。Linux 環境では cc -c myprog.c のように、-c オプション付きで cc を実行すると、この myprog.o ファイルを作成できます。

[3]プログラムファイル myprog が作成されると、myprog.o は自動的に削除されます。

[4]cc コマンドのことを「C コンパイラ」と呼びますが、実際には、C 言語で書かれたソースファイルを機械語に翻訳する「コンパイル」の作業と、その結果できたオブジェクトファイルを C ランタイムオブジェクトなどと結合する「リンク」の作業の 2 つを行っています。

[5]前回配布資料の「mmap 用領域」の説明を参照してください。

使って、このプロセスのアドレス空間に取り込みます。myprog の main の中で、printf や scanf の呼び出しを行う call 命令[6]の分岐先は、これらのライブラリ関数が取り込まれた仮想アドレスに書き換えられます。この書き換えは、プログラムの起動時にまとめて行うこともあれば、とりあえず call 命令の分岐先を、この書き換え作業を行う関数にしておき、関数が呼ばれた際に、本来呼び出されるべき関数へ仲介するとともに、同じ call 命令の次の実行に備えて、分岐先の書き換えを行うこともあります。

動的リンカは、さらに、使用されているライブラリ関数がデータを記憶するため必要なメモリ領域 (ページ) を確保するなどして、ライブラリ関数が動作するための環境を整えます。動的リンカが行う以上の作業を**動的 (dynamic) リンク**と呼びます。その後、動的リンカは、myprog のプログラムファイル内に指定されている開始アドレス[7]に分岐して、myprog の実行を開始します。

メモ

**静的リンク**　動的リンクでは、プログラムファイルの実行 (開始) 時に、必要なリンク処理を行ないますが、これに対して、プログラムファイルを作成する時にリンク処理を済ませてしまうことを**静的 (static) リンク**と呼びます。例えば、先ほどの myprog.c のコンパイル時に、cc -static -o myprog myprog.c のように -static オプションを付けて cc コマンドを実行すると[8]、myprog.c に書かれた C プログラムをコンパイルして得られたオブジェクトファイル myprog.o と、そこで使用されている関数 printf や scanf の機械語プログラムがリンクされてプログラムファイル myprog が作成されます。作成された myprog には、関数 printf や scanf の機械語プログラムが含まれることになります。

静的リンクを行うと、出来上がるプログラムファイルは大きくなってしまいますが、実行時にリンクを行う必要がなくなり、その分だけ素早く起動できるようになります。また、プログラムファイル作成後に行われたライブラリのバージョンアップなどによって生じる不具合を減らすことができますが、逆に、プログラムファイル作成時には存在していたライブラリの不具合が、ライブラリのバージョンアップで解決されることも期待できなくなります。

情報処理システム Ⅱ ・第 8 回・終わり

[6]前回配布資料参照。

[7]第 4 回配布資料を参照してください。通常、開始アドレスは、C ランタイムオブジェクト内の機械語プログラムのアドレスとなります。

[8]静的リンクを指示するオプションの書き方はコンパイラによって異なります。