情報処理 (計算機) システム Ⅱ ・第 4 回

2024 年 5 月 10 日

今回の内容

4.1 プログラムファイル

カーネルが起動するユーザプログラム (機械語のプログラム) は、SSD やハードディスクなどの補助記憶装置にファイル[1]として格納されていますが、このファイルはつぎのような情報を含んでいます。

(1) 機械語プログラム自身

(2) 機械語プログラムが使用するメモリ空間やアドレス空間の設定 (どのアドレスに機械語プログラムを読み込むべきか、どの範囲のアドレスをデータ領域[2]やスタック領域[3]として確保すべきかなど)

(3) 機械語プログラムが使用するデータ領域の初期値

(4) 機械語プログラムの実行を開始するアドレス

(5) 機械語プログラム内で、ライブラリ中の手続き (関数) を呼び出している場所 (アドレス) や、呼び出している手続き (関数) の名前とそれを含むライブラリの名前 (パス名) など

また、通常、ファイルの先頭部分には、これらの情報がそれぞれファイルのどの位置に記録されているかを示す情報が格納されます。この部分を**プログラムヘッダ**と呼びます。

メモ

[1] たとえば、Linux 環境の cc などで作成されるオブジェクトプログラムのファイルがそうです。

[2] その機械語プログラムが各種のデータを記憶するために使用する領域です。C プログラムの静的変数 (グローバル変数など) は、ここに記憶されます。

[3] サブルーチン (C 言語の関数など) の呼び出しの際の戻り先のアドレスの記憶や、サブルーチンが一時的に使用する記憶領域 (C 言語の自動変数など) を確保するために使用します。

4.2 プロセス

カーネルが起動すると、まず最初に、いくつかのユーザプログラム[4]を起動します。他のユーザプログラムは、すべて、これらのプログラムのプロセスや、そこから派生した (ユーザプログラムの) プロセスが、カーネルに依頼する[5]ことで起動されます。

プログラムが起動されると、それが新しいプロセスとなる場合もあれば、すでに存在するプロセスの仕事の続きとして、そのプログラムが実行される場合もあります。前者の場合は、まず、その新しいプロセスを管理するために必要な情報を記憶する領域がカーネル内に確保されます。後者の場合は、すでに存在しているプロセス管理のための情報が更新されます。この情報は**プロセス管理情報**と呼ばれ、通常、次のような情報が含まれます[6]。

- プロセスを識別する番号 (プロセス ID)
- プロセスを起動したユーザを識別する番号 (ユーザ ID)
- 実行中のプログラムファイル
- ファイルや入出力装置に対する読み書きの状況[7]
- このプロセスのアドレス空間の設定[8]

メモ

その後、カーネルは以下のような手順でプログラムを起動します。

1. カーネルは、起動するファイルのプログラムヘッダを主記憶装置 (メモリ) 上に読み込みます。
2. カーネルは、プログラムヘッダの情報に基づき、必要なメモリ領域を主記憶装置に確保し、そこに機械語プログラム自身やそのデータ領域の初期値などを、ファイルから読み込みます[9]。また、PC などで利用されるある程度高度な OS の場合では、CPU のアドレス変換機構に対して、仮想アドレスと物理アドレスの対応関係の設定を行います。

[4] これらのユーザプログラムは、通常はサーバプログラムとして動作します。

[5] ユーザプログラムは、システムコールを行うことで、カーネルに対して、プログラムの起動を依頼することができます。

[6] Windows や MacOS、Linux などの OS では、この他にも、そのプロセスのカレントディレクトリ、プロセスの起動時刻、プロセスが CPU を使用した時間 (CPU 時間) の総計、プロセスが使用可能なメモリや CPU 時間の総量など、多くの情報がプロセス管理情報に含まれます。

[7] たとえば、そのプロセスがファイルに対する読み書きを行っている場合、どのファイルのどの位置を読み書きしているのかが保持されます。

[8] 詳しくは、次回以降に解説します。

[9] (この科目の後半で解説する) デマンドページング方式の仮想記憶システムが採用されている OS では、プログラムファイルを一度にすべて読み込むのではなく、プログラムの実行中に、必要に応じて必要な部分が主記憶装置に読み込まれます。

3. カーネルは、CPU を特権モードから非特権モードに移行させて、主記憶装置に読み込んだ機械語プログラムの実行を開始します[10]。これは、ここでカーネルプログラムの実行が一旦終了することを意味します。後述する**割り込み**や**例外**と呼ばれる事象が発生したり、ユーザプログラムがカーネルの提供している機能 (システムコール) を呼び出すまでは、CPU は非特権モードでユーザプログラムの実行を続けます。起動されたユーザプログラムは、カーネルからの干渉を受けずに独自に動作します。

メモ

4.3 コンテクストスイッチ

CPU は限られた数しかありませんから、その数を越えるプロセスを同時に実行することはできません。より多くのプロセスを並行して実行するため、多くのカーネルには、CPU の割り当て先となるプロセスを短い時間間隔で切り替える仕組みが備わっています。また、ユーザプロセスが、カーネルの管理下に置かれているさまざまな資源にアクセス (システムコール) する際には、CPU は特権モードに移行するとともに、カーネル内のプログラムの実行に切り替わります。

このように、あるプログラムの実行中に、そのプログラムの実行を一時中断して、異なるプログラムの実行に移行することを**コンテクストスイッチ**(context switch) と呼びます。コンテクストスイッチが行われる際には、実行中のプログラムは一旦中断され、その時点の CPU の状態が (カーネル内に) 記憶されます。この CPU の状態のことを**コンテクスト**と呼び、つぎのような内容が含まれます。

- CPU のプログラムカウンタの値
- CPU 中の各レジスタの値
- CPU 中の各状態フラグの値
- CPU のアドレス変換機構の設定 (アドレス空間の割り当て)

一時停止状態にあるプログラムの実行を再開する場合には、この情報に基づいて CPU の状態を復元します。

メモ

[10] つまり、プログラムヘッダに記録されている開始アドレスに分岐 (ジャンプ) します。

4.4 マルチタスキング

CPU を割り当てるプロセスを次々と切り替えるようにして、CPU の数を越えるプロセスを並行して実行させる手法を**時分割処理** (time sharing) と呼びます。また、時分割処理を行うことで、複数のプロセスを並行して動作させることを**マルチタスキング**[11]、マルチタスキングが行える OS を**マルチタスク OS**と呼びます。このとき、CPU の割り当て先となるプロセスを切り替える方法には、大きく分けると次の 2 通りがあります。

メモ

プリエンプティブマルチタスク (preemptive multitasking)　一つのプロセスの実行が一定の時間 (たとえば 10ms) 経過したら、強制的にその実行を中断し、他のプロセスに CPU を割り当てます。PC やスマートフォンなどで使用されるような、ある程度高度な OS で採用されるマルチタスキング方式です。ユーザプログラムは、マルチタスキングが行われることを特に意識して作成されている必要はありません。

ユーザプロセスの実行が一定の時間経過したことをカーネルが知るためには、CPU 内や、その周辺回路として用意されるタイマーを使用します。このタイマーは、設定された時間が経過した時点で、CPU に対して「割り込み」と呼ばれる信号を送ります。「割り込み」を受けた CPU は、カーネルによってあらかじめ設定されている処理を行いますが、そこで CPU を割り当てるプロセスが切り替えられます。割り込みとその処理については、節を改めて後ほど説明します。

メモ

協調的マルチタスク (cooperative multitasking)　タイマーなどを使用せず、各プログラムが自発的に CPU の割り当てを放棄することでコンテクストスイッチを行います。各プログラムは特定のシステムコールを行うことで、カーネルが CPU を割り当て直す機会を作ります。ユーザプログラムは、そのことを意識してプログラミングされていなければなりません。不具合などによりプログラムが暴走してしまうと、カーネルがプロセスを切り替える機会が失われてしまいますので、その影響がすべてのプロセスの実行に及んでしまい、システム全体が機能しなくなってしまうこともあります。協調的マルチタスクは、1990 年代までの PC 向けの OS で採用されていました

[11] マルチプログラミング と呼ぶこともあります。

が、現在では、家電製品などに組み込まれる非常に単純な OS か、機械の制御など、ハードウェアからの信号に迅速に対応することが必要となるシステムなどに限って採用されています。

メモ

4.5 プロセスの状態遷移

マルチタスキングを行うと、プロセスが入力装置 (マウスやキーボードなど) や補助記憶装置 (ハードディスクなど) から読み出されるデータを待っている間にも、順番待ちをしている他のプロセスに CPU を割り当てることができますので、CPU という資源を効率よく利用することができます。

カーネルが、各プロセスへ CPU を割り当てる作業を**スケジューリング**と呼びます。各プロセスは、下の図 1 のように、**実行可能 (実行待ち)**、**実行中**、**イベント待ち**の 3 つの状態を移っていきます。同時に「実行中」の状態になることのできるプロセスの数は、その計算機に搭載されている CPU の数を越えることはありません。

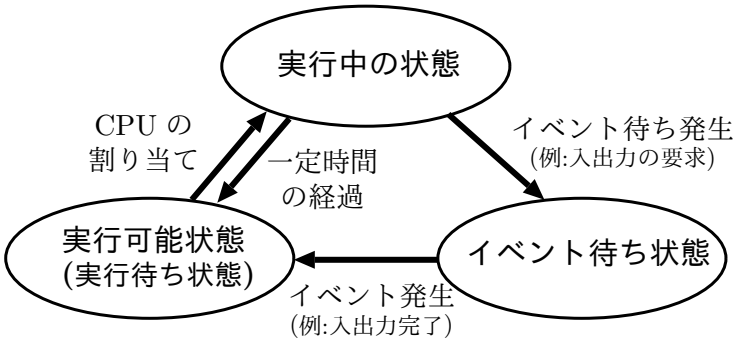

図 1: プロセスの状態の遷移

メモ

4.6 割り込み

CPU は機械語プログラムを逐次実行していきます。基本的にその流れは、実行中のプログラム自身 (プログラム中に置かれた分岐命令) によって制御されますが、入力装置や補助記憶装置から読み込もうしたデータが到着した、アクセスしたメモリアドレスに物理的なメモリが割り当てられていなかったなど、プログラムの基本的な実行の流れの外で発生する出来事 (イベント) を処理する

仕組みが CPU に備わっています。このような出来事は、ハードウェアに起因するものと、ソフトウェアに起因するものの 2 種類があり、これらをまとめて (広義の)**割り込み**と呼びます。

割り込みを検知した CPU は、自動的に現在のコンテクスト[12]を保存し、特権モードに移行するとともに、カーネルによってあらかじめ設定されているアドレスに分岐します。そこでカーネルは、割り込みの処理を行うことができます。割り込みの処理が終わると、カーネルは、保存しておいたコンテクストを復帰させ、中断しておいたプログラムを再開します。

メモ

ハードウェア割り込み　CPU には、その外部から信号を与えることにより、現在実行中のプログラムの処理を中断して、別の処理を行わせるための機構が備わっており、この信号のことを割り込み信号と呼びます。プリエンプティブマルチタスキングで紹介したように、タイマーに設定された時間が経過したときや、入力装置や補助記憶装置からのデータが読み込み可能になったり、出力装置 (や補助記憶装置) がデータを受け取る準備ができたときに、これらの装置が CPU に対して割り込み信号を送ります。このような、ハードウェアに起因する割り込みのことを**ハードウェア割り込み**と呼びます。ハードウェア割り込みのみを (狭義の)**割り込み**とする場合もあります。

メモ

ソフトウェア割り込み　ハードウェア割り込みは CPU の外部からやってきますが、CPU が実行しているプログラム (ソフトウェア) 自身によって予期しない事象が引き起こされる場合もあります。これをハードウェア割り込みと区別して、**ソフトウェア割り込み**と呼びます。ソフトウェア割り込みには、0 を除数として割り算を行ったり、メモリが割り当てられていないアドレスに対してメモリアクセスを行ったりした場合など、通常の機械語命令の実行によって予期せず引き起こされるものと、カーネルに対するシステムコールを実現するために CPU に用意されている特定の機械語命令の実行により意図的に引き起こされるものとがあります。前者のソフトウェア割り込みは、特に**例外**[13]と呼ばれます。

情報処理 (計算機) システム Ⅱ ・第 4 回・終わり

[12]必要最小限のコンテクストが保存され、残りの部分は、特権モードへの移行後にカーネルプログラムによって保存されます。

[13]トラップ (Trap) と呼ぶこともあります。