情報処理システム Ⅱ ・第 9 回

2025 年 6 月 6 日

配布資料の内容

9.1 ファイル管理

一般に、ハードディスクや SSD などの補助記憶装置に格納されたデータのまとまりの 1 つ 1 つを**ファイル**と呼びます。ソースプログラムや機械語プログラム、文書、画像ファイルなど、いろいろなデータがファイルとして記憶されますが、それぞれのファイルは、ある長さのビット列に過ぎません。

1 つのファイルの内容は、オペレーティングシステム (カーネル) によって、ハードディスクや SSD、DVD、フラッシュメモリなどの補助記憶装置に格納されます。アプリケーションプログラムは、カーネルのシステムコールを呼び出すことで、この内容を読み書きします。DVD-ROM のような読み込み専用の記憶装置 (媒体) にファイルが置かれている場合は、当然、その内容を変更することはできませんが、ハードディスクや SSD、フラッシュメモリのように、書き込みもできる記憶装置の場合は、すでにあるファイルの内容を書き換えたり、ファイルを削除したり、あるいは、新しくファイルを作成したりすることができます。

メモ

ファイルのメタデータ　1 つのファイルが持つ情報の内、最も重要な部分はそのファイルの内容 (ビット列) 自身ですが、多くの OS で、これに加えて**メタデータ**(metadata) と呼ばれる次のような情報が各ファイルに結び付けられて記憶されます。

- ファイルの大きさ (バイト数)
- ファイルの所有者 (のユーザ ID)
- ファイルのアクセス保護情報
- ファイルの最終更新日時
- ファイルの最終参照日時

ファイルの名前空間　計算機の補助記憶装置にはたくさんのファイルを作成することができますが、これらを区別するために、それぞれファイルには、それを指し示すための名前 (文字列) が付けられるようになっています。1 つのオペレーティングシステムにおいて、ファイルを指し示すため

に使われる名前 (文字列) 全体を、ファイルの**名前空間**と呼びます。名前空間に属する文字列[1]が、それぞれどのように特定のファイルを指し示すのかの決まりが定められています。

最も単純な名前としては「レポート `1.docx`」とか「`test.c`」のようなものが考えられますが、補助記憶装置には膨大な数のファイルを作ることができますので、このような単純な名前だけで、すべてのファイルを区別するのは困難です。そこで、補助記憶装置に記憶されているファイルをグループ化し、「どのグループのどのファイル」というような指し示し方ができるようになっているのが普通です。この「グループ」は、一般に**ディレクトリ**と呼ばれ、ファイルの「置き場所」として働きます。異なるディレクトリに置かれているのであれば、異なる 2 つのファイルが同じ「`test.c`」という名前を持つことができます。ディレクトリは「フォルダ」と呼ばれることもあります。

多くのオペレーティングシステムでは、1 つディレクトリの中に、さらに別のディレクトリを置くことができるようになっています。1 つのディレクトリに対して、そのディレクトリが置かれているディレクトリのことを**親ディレクトリ**と呼び、逆に、1 つのディレクトリに対して、そのディレクトリに置かれているディレクトリのことを**子ディレクトリ**、あるいは**サブディレクトリ**と呼びます。1 つのディレクトリの子ディレクトリは複数ある場合もありますし、まったく無い場合もあります。

メモ

パス名　図 1 は、Linux におけるディレクトリ階層 (一部) の例です。すべてのファイルやディレクトリの祖先となっているディレクトリは**ルートディレクトリ**と呼ばれます。ディレクトリの構造が階層的な場合、ファイルの名前空間も階層的な構造を持つことになります。このとき使われる文字列は「... というディレクトリに置かれた ... というディレクトリに置かれた ... というディレクトリに置かれた ... というファイル」いったような意味を持っており、一般に**パス名**と呼ばれます。

「パス名」は、ファイルやディレクトリの住所の書き方ようなものです。Linux では、`/` で始めて、ルートディレクトリを起点として目的のファイルやディレクトリに至るまでのディレクトリの名前を `/` で区切って並べて「`/usr/bin/ls`」のように書いた**絶対パス名**[2]や、カレントディレクトリ (そのプロセスが動いているディレクトリ) を起点として、そこから目的のファイルやディレクトリに至るまでのディレクトリの名前を `/` で区切って並べた「`bin/ls`」のように書かれる**相対パス名**[3]が用いられます。あるプロセスのカレントディレクトリの (絶対) パス名が `/usr` である場合、そのプロセスにとっては、`/usr/bin/ls` も `bin/ls` も同じファイルを指すパス名となります。

[1] 後述する「パス名」などのことです

[2] 絶対パス名は必ず `/` で始まります。ルートディレクトリ自身は `/` で表します。

[3] 相対パス名は `/` 以外の文字で始まります。

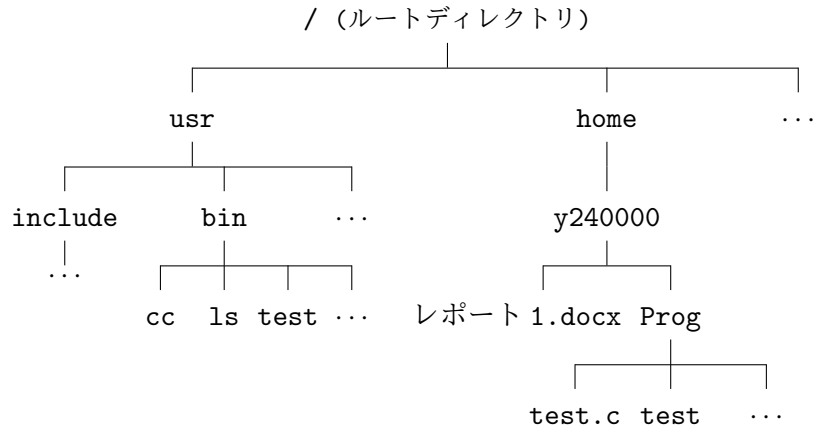

図 1: Linux におけるディレクトリ階層の例

メモ

9.2 ファイルシステム

1 つのファイルのデータは、補助記憶装置 (たとえばハードディスク) の中で、連続した領域を占めるとは限りません。1 つのファイルの内容は、オペレーティングシステムによって、512 B 〜 32 KiB 程度の大きさの**ブロック**と呼ばれる単位に分割され、このブロックを単位として、図 2 のようにいくつかの場所を使って記憶されます。

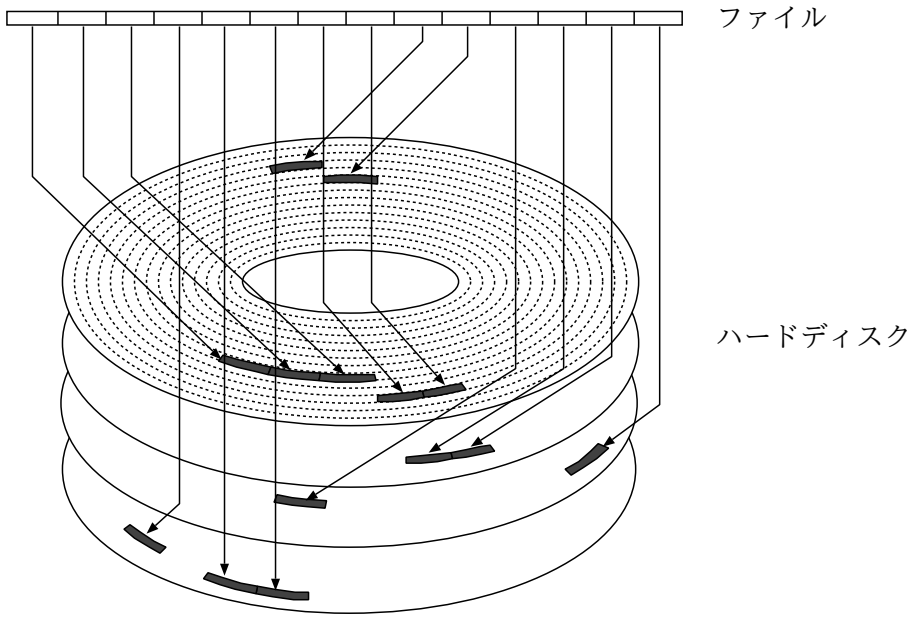

図 2: ハードディスク中で 1 つのファイルが占める領域の例

このためオペレーティングシステムは、どのファイルのデータのどの部分が (ハードディスクなどの) 補助記憶装置のどこに置かれているのかを知っていなければなりません。また、それぞれのディレクトリにどのような名前のファイルが置かれているかが分からないと、与えられたパス名によって指し示されたファイルの記憶場所に辿り着くことが出来なくなってしまいます。このような (ファイル管理のための) 付加的な情報も、一定の約束事に従って補助記憶装置の一部に格納されます。

メモ

各ファイルのデータ (やメタデータ) を補助記憶装置にどのような仕組みで記憶するのかについて約束事、あるいはその約束事に従って補助記憶装置の中に構築されたデータの構造を**ファイルシステム**と呼びます。ハードディスクや SSD などの補助記憶装置の全体、あるいはその一部を、ファイルシステムの約束事にしたがって初期化する (決まったデータ構造を構築する) ことを「**(論理) フォーマットする**」と言います。補助記憶装置を使ってファイルを記憶するためには、まずフォーマットを行って、そこにファイルシステムを構築することが必要ですが、このとき (多くの場合) 補助記憶装置 (の該当部分) に記憶されていたデータは上書きされて消えてしまいますので注意が必要です。

1 つのオペレーティングシステムでも、(状況に応じて) 複数の方式のファイルシステムを使用することがあります。Windows の FAT32[4] や NTFS、Linux の ext4 は、このようなファイルシステム (の方式) に付けられた名前です。ファイルシステムが違えば、そこに記憶されているファイルの読み書きのやり方が異なってきますので、そのファイルシステムに対応しているオペレーティングシステムでないと、補助記憶装置に記録されたファイルの内容を読み書きすることはできません。

メモ

断片化　ハードディスクなどの補助記憶装置の構造上、ファイルを可能な限り連続したブロックに記憶する方が、より高速にファイルの内容全体を読み書きできるようになります。しかし、アプリケーションプログラムの指示によって、多くのファイルが作成、削除されたり、各ファイルの大きさが変化することで、どうしても補助記憶装置の記憶領域は虫食い状態となり、必ずしも連続した領域を 1 つのファイル全体のために使用することができなくなります。オペレーティングシステ

[4] USB フラッシュメモリでよく用いられます。

ムは、できるだけ連続した領域を 1 つのファイルに割り当てようとしますが、これが困難な場合は (図 2 のように) いくつかのブロックに分割して 1 つのファイルの内容を記憶するわけです。

ファイルの内容が細く分割されて、(ハードディスクなどの) 補助記憶装置の記憶領域内に散らばってしまうことを**断片化 (フラグメンテーション)**と呼びます。たくさんのファイルが作られて、補助記憶装置の記憶容量が残り少なくなった状態で、ファイルの作成や削除が繰り返されたり、ファイルの大きさの変更が繰り返されると断片化が起きやすくなりますが、その程度は、使われているファイルシステムが何であるかによって違ってきます。たとえば、FAT32 は、NTFS や ext4 に比べると断片化が起きやすいという性質を持っています。断片化がひどくなると、1 つのファイル内容を読み書きするためにハードディスクのあちこちにアクセスする必要が出てきますので、ファイルの読み書きにより時間が掛かるようになります。

メモ

9.3 Unix 系 OS のファイルシステム

Linux などの Unix 系 OS では、それぞれ特有のファイルシステムが使用されていますが、これらは共通の特徴を持っています。

inode AT&T 社の Unix のファイルシステムから派生した Unix 系 OS のファイルシステム[5]では、1 つのファイルは、**inode** と呼ばれる情報と、そのファイルの内容 (中身のデータ) に関する情報の組で表現されます。どちらも補助記憶装置のいくつかのブロックを使用して記憶されます。1 つの inode は、そのファイルのメタデータの情報と、そのファイルの内容 (中身のデータ) を格納しているブロック群がどこに位置しているかに関する情報で構成されています。inode はファイルシステム内で一意となる inode 番号で識別され、inode 自体にはファイルの名前に関する情報は含まれていません。ファイルシステム内のファイルは、この inode 番号で識別されます。

メモ

ディレクトリ ファイルの置き場であるディレクトリは、ファイルシステム中の特殊なファイル[6]として表現され、通常のファイルとは異なるものとして扱われます。1 つのディレクトリの内容 (中

[5]Solaris の UFS、FreeBSD、NetBSD、OpenBSD の UFS2、Linux の ext4 など。

[6]ディレクトリもファイルですので、それを表す inode が存在します。

身のデータ) は、そのディレクトリに置かれたファイルの名前と、そのファイルの inode 番号との対応表です。この対応表に記されたものがファイルの名前に相当するものになります。後述のように 1 つのファイルを複数のディレクトリに置くことができますので、1 つのファイルが複数の名前を持つことがあります。

例えば、あるディレクトリに `foo.c`、`foo` という 2 つのファイルと `bar` というサブディレクトリが置かれていたとすると、このディレクトリは右のような表に相当する情報を保持します。

名前	inode 番号
`.`	6712
`..`	342
`foo`	8729
`foo.c`	1203
`bar`	2312

表中の「`.`」と「`..`」は、すべてのディレクトリに自動的に記録されるもので、それぞれ、「自分自身のディレクトリ」と「自分の親ディレクトリ」を意味し、これらをパス名の一部に使用することができます。

メモ

ハードリンク　ディレクトリに記録されている、ファイル名から inode への対応を**ハードリンク** (hard link) と呼びます。ディレクトリは、単にファイル名と inode の対応表でしかありませんから、1 つのファイルが複数のディレクトリに置かれることも可能です[7]。inode には、メタデータの一部として、そのファイルへのハードリンクの数が格納されています。ファイルが新たに生成される際には、必ず 1 つのディレクトリからハードリンクされた状態となりますが、`link` というシステムコールを使って、すでに存在しているファイルを別のディレクトリからハードリンクすることができます[8]。逆に、`unlink` というシステムコールで、ファイルへのハードリンクを削除することができます。inode に記録されたハードリンクの数が 0 になると、そのファイルは削除され、inode とデータを格納していたブロック群が解放されます。

メモ

[7] AT&T 社の Unix に由来するファイルシステムでは、ディレクトリもファイルの一種ですが、通常、1 つのディレクトリが複数のディレクトリに置かれる (複数のディレクトリのサブディレクトリとなる) ことは許されていません。一方、macOS の HFS+ のように、これを許すファイルシステムもあります。

[8] ただし、単一のファイルシステム内でしかハードリンクすることはできません。

シンボリックリンク　Unix 系 OS のファイルシステムには、ハードリンク以外にも、**シンボリックリンク** (symbolic link) と呼ばれるものを作成して、特定のディレクトリにファイルを置くことができます。シンボリックリンクは特殊なファイルとして表現されており、そこにはリンク先のファイルのパス名が記録されています。たとえば、ユーザプログラムがシンボリックリンクのファイルを読み書きしようとすると、カーネルは、そのシンボリックリンクに記録されたパス名を読み込み、実際にはそのパス名で指定されるファイルを読み書きしようとします[9]。

ハードリンクでは inode 番号でファイルを指定しますので、同じファイルシステム内のファイルしかリンクすることはできませんが、シンボリックリンクでは別のファイルシステム内のファイルへのリンクを作成することができます。また、シンボリックリンクのリンク先は、ディレクトリなど、どのような種類のファイルでも構いません。ただし、シンボリックリンク先が削除されてしまうと、そのシンボリックリンク自身も削除されたのと同じ状態になってしまいますので注意が必要です。

情報処理システム Ⅱ ・第 9 回・終わり

[9] そのファイルもやはりシンボリックリンクであれば、さらにそこに記録されているパス名を使用します。必要に応じて (一定の数を限度として) このような処理が繰り返されます。