Operating System

2024-08-03

01-6.pdf: 第 6 章 ファイルシステム

1. ファイル編成の順編成と直接編成の違いをまとめよ。

順編成は一定サイズのデータ（レコード）が順次に並ぶ構造であり、アクセスは先頭から順次に行われます。シリアルアクセス記憶装置（テープ装置など）をモデル化しています。直接編成はデータへのアクセスの順番に関係する構造がなく、任意の位置へのアクセスが可能です。ランダムアクセス記憶装置（磁気ディスク装置など）をモデル化しています。

2. Unix のファイルシステムの特徴をまとめよ。

Unix のファイルシステムは単純で柔軟です。ファイルは 1 バイト単位での順編成を基本とし、シーク操作によって直接編成としても使用できます。ファイルサイズは伸縮自在であり、OS が制御します。また、入出力処理が統一化されており、プログラムから見て入出力装置への入出力とファイルへの入出力に差がありません。

3. ディレクトリの役割を述べよ。

ディレクトリはファイルを分類・整理するための構造です。記憶装置内部に存在するファイルの名前を一覧情報として保持・提示します。内部的にはファイルの名前と場所などの対応表として保持されます。

4. 階層的なディレクトリを用いる利点を述べよ。

階層的なディレクトリ構造を用いることで、ファイルを体系的に整理できます。分類項目の下をさらに細分することが可能であり、ファイルの体系的な管理が容易になります。

5. カレントディレクトリに関して説明せよ。

カレントディレクトリは現在開いているディレクトリであり、ここを起点としてファイルや他のディレクトリを指定できます。ユーザの操作で移動可能であり、主に CUI で現在作業を行っているディレクトリのことを指します。

6. 絶対パス名と相対パス名の違いをまとめよ。

絶対パス名は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経路を指定します。相対パス名はカレン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経路を指定します。

7. UNIX と Windows のファイルシステムの構造の違いを説明せよ。

UNIX のファイルシステムはシステム全体のファイルを一つの木構造として管理し、ルートディレクトリが一つだけ存在します。一方、Windows ではファイルを格納する外部記憶装置をドライブと呼び、ファイルの木構造がドライブごとに存在します。各ドライブの先頭のディレクトリがルートディレクトリです。

8. マウント、アンマウントはどのような操作か説明せよ。

マウントは異なるボリュームを接ぎ木して単一の木構造を維持するための処理であり、あるシステムの任意のディレクトリを他のシステムの任意の場所へマウントすることが可能です。アンマウントはマウントされたファイルシステムを取り外す処理です。

9. i-node で索引表が一段しかなかったらどのような不具合が生じるか検討せよ。

i-node の索引表が一段しかない場合、非常に大きなファイルを扱うことが難しくなります。一段の索引表では管理できるブロックの数が限られてしまい、大容量のファイルを保存するためには複数の索引表を使用する必要があります。

10. i-node に含まれる管理情報にはどの様な情報があるかまとめよ。

i-node には以下のような管理情報が含まれます：

- ファイルの所有者
- ファイルタイプ
- ファイルサイズ
- ファイルのアクセス許可情報
- 前回のファイルアクセス/更新時間
- 各ブロックの保存場所の情報（直接、間接などのエントリ）

11. (参考) i-node にファイル名が含まれていない事の利点・欠点を検討せよ。

利点：

- ファイル名の変更が容易になる。
- 同一 i-node で複数のファイル名を持つことが可能になる（ハードリンク）。

欠点：

- ファイル名の管理が別の構造（ディレクトリ）で行われるため、管理が複雑になる。
- ファイル検索の際に i-node とディレクトリ情報を照合する必要がある。

12. (参考) ファイルを保存する際に、連続したブロックに保存しないことの利点・欠点を検討せよ。

利点：

- ファイルサイズの変更に柔軟に対応できる。
- ファイルシステムの空き領域を効率的に利用できる。

欠点：

- データの断片化が発生しやすくなるため、アクセス速度が低下する可能性がある。
- ファイルの読み書き時に複数の場所にアクセスする必要があるため、処理が複雑になる。

13. (参考) ブロックサイズを小さく（または大きく）することの利点・欠点を検討せよ。

小さくする利点：

- 小さなファイルを効率的に保存できる。
- ディスクスペースの無駄が減る。

小さくする欠点：

- 管理すべきブロックの数が増えるため、管理コストが上がる。
- 大きなファイルの読み書き速度が低下する可能性がある。

大きくする利点：

- 大きなファイルの読み書き速度が向上する。
- 管理すべきブロックの数が減るため、管理が簡単になる。

大きくする欠点：

- 小さなファイルを保存する際にディスクスペースの無駄が増える。
- データの断片化が発生しやすくなる。