8. Linux のシステムプログラミングに関する知識 I

1. 科目の概要

Linux においてシステムが提供するリソースを活用してプログラムを実行するシステムプログラミングについて解説する。プログラムの作成方法から始まり、プログラムからのファイル操作、パラメータの受け渡し、ライブラリの利用、メモリ管理やデータベースの利用までを説明する。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説　明	シラバスの対応コマ
I-8-1. エディタを用いたソースファイルの作成	Linuxのシステムへログインし、エディタを起動してソースファイルを作成するまでの手順を説明する。また代表的なエディタであるviとemacsについて、基本的な操作方法を解説する。	1,4
I-8-2. コンパイラの仕組みと実行ファイルの作成	C言語プログラムを題材として、コンパイラの仕組み、プリプロセッサ、コンパイラ、リンカの役割を解説する。またソースプログラムからオブジェクトファイルを生成し、実行プログラムを構築するまでの手順やコンパイラオプションの指定などを説明する。	1
I-8-3. シェルスクリプトによる定形処理	シェルの種類やシェルスクリプトの概念、対話的シェルの操作方法を説明する。スクリプトを作成してから実行するまでの手順を示し、また基本的なシェルコマンドやシェルスクリプトの文法、記述方法について言及する。	1,2
I-8-4. 低水準ファイル処理によるファイル操作	低水準ファイルアクセスを行うシステムコールを使ってファイルを操作する方法を示す。ファイル操作のモード、ファイルの生成、オープンとクローズ、ファイルの読み書きとファイルパーミッションの変更、その他ファイル操作に必要なシステムコール群について説明する。	3
I-8-5. 標準入出力ライブラリによるファイル操作	標準入出力ライブラリを用いてファイルを操作する方法を示す。低水準ファイル処理のそれぞれに対応する各関数を紹介し、さらに書式付き入出力や利用上の留意点も示す。また一時ファイルを作成する関数についても触れる。	3
I-8-6. ファイルシステムの操作	ファイルシステムの概念や構造、ファイルシステムの種類と各ファイルシステムの特徴について説明し、複数のディスクブロックをひとつのツリーとして扱う考え方とその方法を解説する。またディレクトリをプログラムから操作する方法や特殊なディレクトリについて説明する。	4
I-8-7. コマンドラインオプションと環境変数のプログラムへの渡し方	シェルからプログラムへパラメータを渡す方法として、コマンドライン引数による方法と環境変数を用いる方法を説明する。渡されたパラメータをC言語プログラム内部で処理する方法やプログラムからの返り値の取扱いについて解説し、代表的な環境変数についても説明する。	5
I-8-8. ライブラリとプログラムの関係	プログラムをコンパイルしてできあがる実行ファイルと、実行ファイルから関数呼び出しで利用するライブラリの関係を説明する。また共有ライブラリの考え方と、実行時の動作、lddコマンドによる利用ライブラリの一覧やsizeコマンドによるメモリ利用状況の把握などを解説する。	5
I-8-9. ライブラリの種類とライブラリの利用方法	ライブラリにはstaticライブラリとsharedライブラリの2種類があることを説明し、それぞれの利点と特徴、利用方法について解説する。また、ライブラリを利用したシステムプログラミングの例を示す。ライブラリで提供される関数とシステムコールの違いについての説明も行う。	6
I-8-10. メモリ管理、ファイル管理とデータベースの利用	OSが管理するメモリの動的な利用方法、ファイルの排他処理として行うロックの方法、Berkeleyデータベースの利用など、システムプログラムから大規模なデータを管理する方法について解説する。	7

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念･知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「8. Linux のシステムプログラミングに関する知識Ⅰ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）							応用レベル（Ⅱ）							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8. Linuxのシステムプログラミングに関する知識	<ログイン手順とコンパイル手順>	<shellプログラミング>	<ファイル入出力プログラミング>	<ファイルシステム>	<UNIX環境>	<ライブラリの利用方法と作成手順>	<データの管理>	<ソフトウェアの開発環境>	<デバッグ>	<プロセスとスレッド>	<シグナル>	<プロセス間通信とパイプ>	<端末機器の入出力>	<セマフォ、共有メモリ、メッセージキュー>	<ネットワークプログラミング>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_08.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織間連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要 [8-Ⅰ-4]	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理 [8-Ⅰ-7]	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピューターアーキテクチャ	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CAO0. 歴史と概要	CE-CAO1. コンピュータアーキテクチャの基礎	CE-CAO2. メモリシステムの構成とアーキテクチャ	CE-CAO3. インタフェースと通信	CE-CAO4. デバイスサブシステム	CE-CAO5. CPUアーキテクチャ	CE-CAO6. 性能・コスト評価	CE-CAO7. 分散・並列処理	CE-CAO8. コンピュータによる計算	CE-CAO9. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、Linux という具体的な環境におけるプログラミング手法の知識が含まれる。ファイルシステムやライブラリに関するプログラミング手法について Linux を通して習得することとなる。

科目名	第1回	第2回	第3回	第4回	第5回	第6回	第7回
8.Linux のシステムプログラミングに関する知識Ⅰ	(1)ログイン手順 (2)エディタ (3)コンパイル手順	(1)シェル (2)シェル構文	(1)低水準ファイル処理に関する説明事項 (2)ファイル操作に関する説明事項 (3)標準入出力ライブラリに関する説明事項 (4)書式付き入出力 (5)一時ファイルを作成するときにtmpnam、tmpfileを利用する。	(1)ディレクトリの保守 (2)ファイルシステムの構造 (3)/proc ファイルシステムに保持される情報を理	(1)コマンドラインオプションと環境変数 (2)代表的な環境変数 (3)共有ライブラリの利用とメモリ配置の確認	(1)static ライブラリ (2)shared ライブラリ (3)ライブラリ関数とシステムコールの違い	(1)メモリの管理 (2)ファイルのロック (3)データベース

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-1. エディタを用いたソースファイルの作成	
対応する コースウェア	第1回（ログイン手順とコンパイル手順） 第4回（ファイルシステム）	

I-8-1. エディタを用いたソースファイルの作成

Linux のシステムへログインし、エディタを起動してソースファイルを作成するまでの手順を説明する。また代表的なエディタである vi と emacs について、基本的な操作方法を解説する。

【学習の要点】

* Linux システムプログラミングは、メモリ管理、ファイルシステム、ネットワーク、プロセス、マルチスレッド、プロセス間通信といったカーネルの基本機能を操作するためのプログラム作成することである。
* ソースファイルの作成手順は、サーバにログインした後、ソースファイルを保存するディレクトリに移動する。
* エディタを起動して、ソースコードの入力・編集を行って保存する。

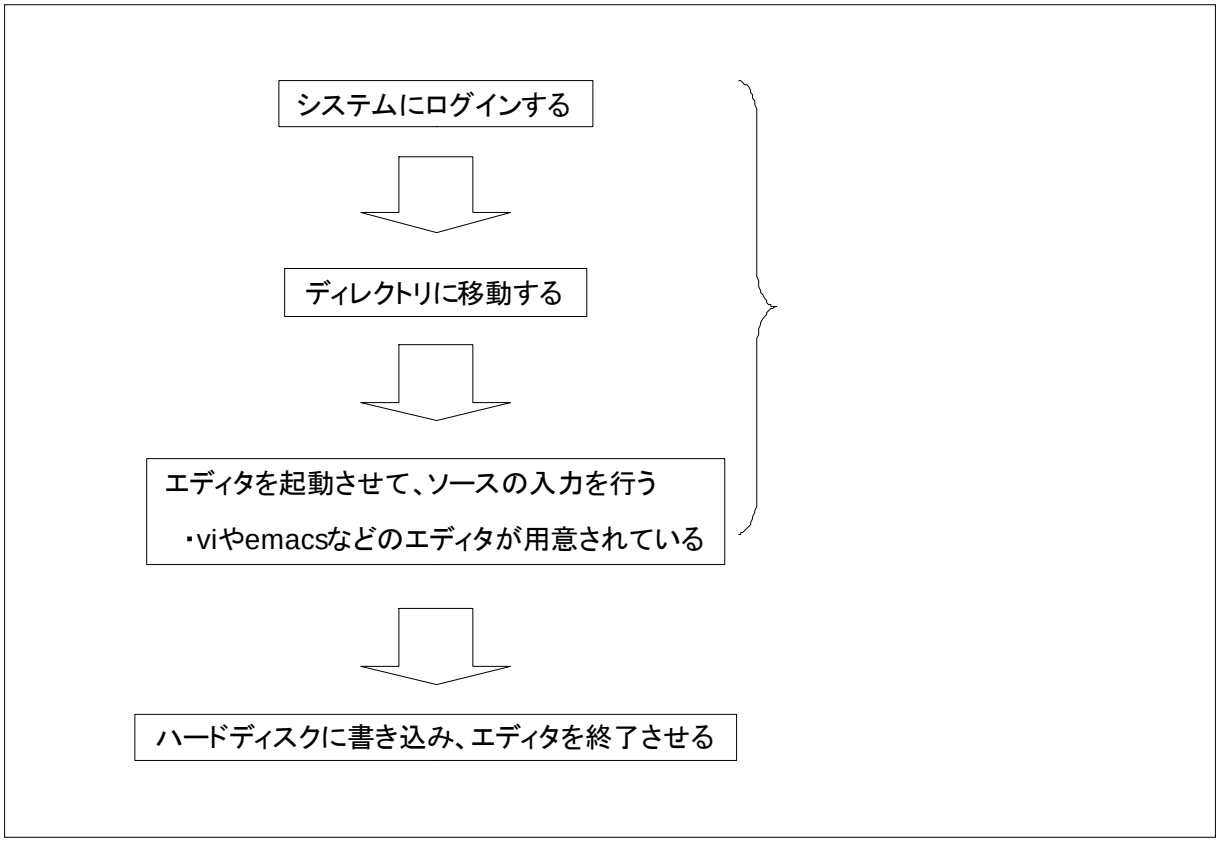

図 I-8-1.　ソースファイルの作成

【解説】

1) ソースファイルの作成手順

新たにソースファイルを作成するための手順を、以下に明記する。

＊ まず最初に、Linux システムにログインする。
システム管理者より受け取った「ログイン名」と「パスワード」を使って、システムにログインする。

＊ ディレクトリに移動する。
Linux ファイルシステムは、ディレクトリという概念でファイルの管理や保存を行っている。そのため、新たに作成したいファイルを保管するためのディレクトリへ事前に移動しておく。

＊ エディタを起動させて、ソースの入力を行う。
ファイルを作成するためには、用意されている強力なエディタを利用する方法が簡単である。

＊ 入力した情報をハードディスクに書き込み、エディタを終了させる。
ソースコードの入力完了後、エディタの機能を使ってハードディスクにソースコードの書き込みを行い、エディタを終了させる。

2) vi (Visual editor)エディタ

vi エディタを起動させる際には、引数として'作成したいソースファイル名'を指定する。

＊ コマンドモード ： このモードでは、以下に示すコマンドが使用可能である。
- a :挿入モードへ切り替える。カーソルの次の文字から文字の挿入が可能。
- i :挿入モードへ切り替える。カーソルの場所から文字の挿入が可能。
- o もしくは O :現在の行の前後に新しい行を追加する。
- r :現在のカーソルの場所の文字を変更する。
- x :現在のカーソルの場所の文字を削除する。
- :(コロン) :ex モードへ切り替える。

＊ 挿入モード ： ソースコード本文も入力を行う。[ESC]キーの押下でコマンドモードに戻る。

＊ ex モード ： 内容を記録したり、vi エディタを終了させたりといったコマンドを実行する。
- wq :作成(修正)した内容をファイルへ書き込んでから、エディタを終了させる。
- q! :作成(修正)した内容は書き込まずに、エディタを終了させる。

3) emacs

emacs には語や段落の操作やソースコードを読み易くする構文の強調機能や、利用者の定義する一括編集コマンドを動かす「キーボード・マクロ」の実行のための機能など、多数の機能がある。

＊ 主モード ： このモードでは、以下に示すキー操作が可能である。
- [Ctrl]+v:次の画面に進む。
- <ESC>+v:前の画面に戻る。
- [Ctrl]+p, b, n, f:カーソルを移動させる。
- [Ctrl]+k:行の削除、連続操作で完全な削除
- [Ctrl]+x [Ctrl]+f:ファイルを開く
- [Ctrl]+x [Ctrl]+s:ファイルをセーブする。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-2. コンパイラの仕組みと実行ファイルの作成	
対応する コースウェア	第1回（ログイン手順とコンパイル手順）	

I-8-2. コンパイラの仕組みと実行ファイルの作成

C言語プログラムを題材として、コンパイラの仕組み、プリプロセッサ、コンパイラ、リンカの役割を解説する。またソースプログラムからオブジェクトファイルを生成し、実行プログラムを構築するまでの手順やコンパイラオプションの指定などを説明する。

【学習の要点】

＊ コンパイラとは、実行可能なオブジェクトコードを作成するツールである。

＊ 実際のコード変換処理の前に、プリプロセッサといったツールが実行される。

＊ コードの変換後には、リンカといったツールが実行されて実行ファイルが完成する。

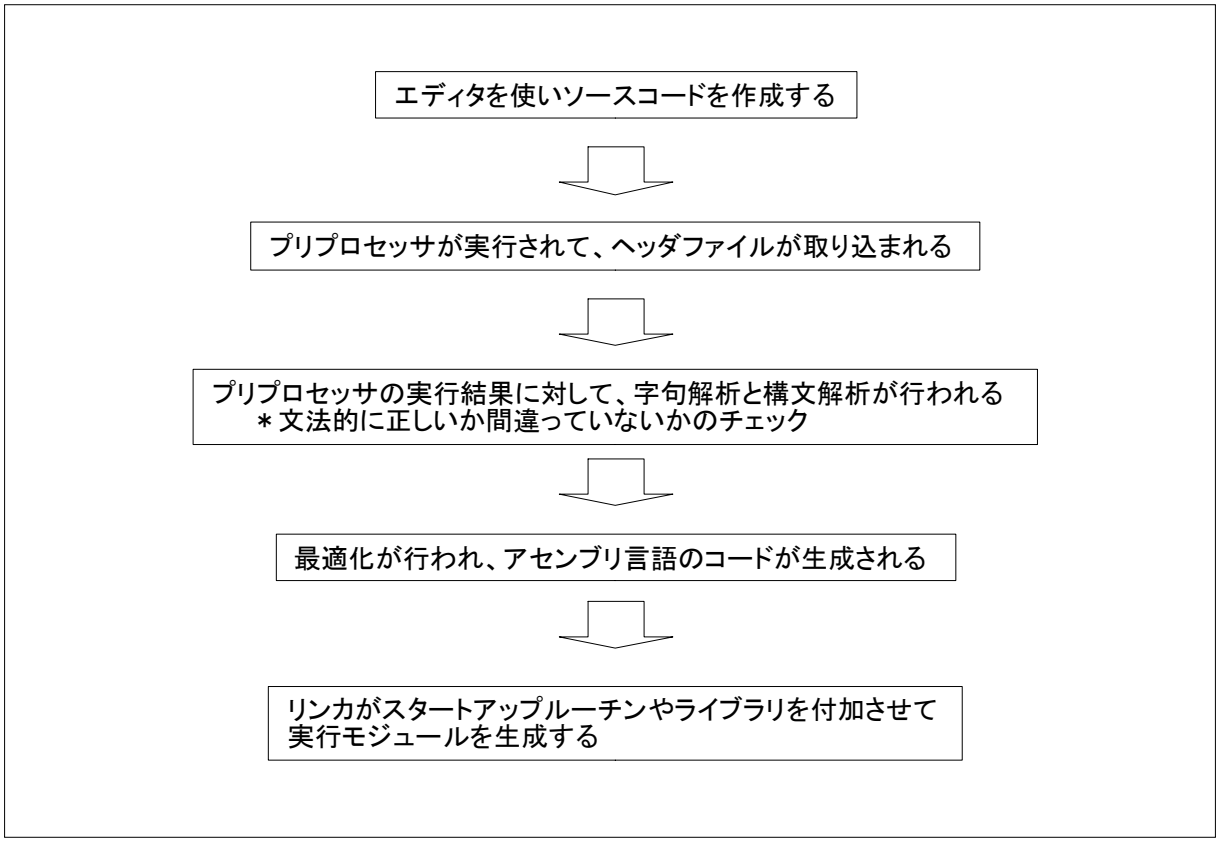

図 I-8-2.　プログラム構築までの手順

【解説】

1) コンパイル時に使用するソフトウェア開発ツール

コンパイラは様々なツールを組み合わせて実行可能なコードを作成する。OSS の GNU gcc コンパイラを使用した場合は、以下に示す機能をまとめて実行することができる。

＊ プリプロセッサ
オブジェクトコードに変換するその前に、ソースコードに一定の規則に従って処理を加えるツール。

＊ コンパイラ
コンピュータ上で実行可能なオブジェクトコードに変換するツール。コンパイラには最適化を行うか行わないかなど、コンパイラの処理方法を制御するオプションがある。

＊ リンカ
オブジェクトコードに、必要なライブラリなどを結合させて、実際に実行することが可能なファイルを生成するツール。

2) コンパイラの仕組み

人間がプログラミング言語の記述ルールに従って作成したソースコードを、コンピュータ上で実行可能なオブジェクトコードに変換することをコンパイルといい、そのためのソフトウェアをコンパイラという。コンパイラの仕組みは以下の通り。

＊ 字句解析
ソースコードに含まれる文字列を、字句という単位に分割する。

＊ 構文解析
文法的に正しいか、構文的に間違っていないかのチェックを行う。

＊ 意味解析
実行することが可能かどうかをチェックする。

＊ コード最適化
実行時により速く実行できるように無駄をなくす処理を行う。

＊ コード生成
オブジェクトコードの生成を行う。

3) 実行プログラム構築までの手順

GNU gcc コンパイラを使用して、C言語のソースコードから実行プログラムを構築する流れを以下に明記する。

＊ vi や emacs などを使ってソースコードを作成する。

＊ gcc コマンドを使って、コンパイルを行う。内部的な処理の流れは以下の通り。
- プリプロセッサが実行されて、#include 文で指定されたヘッダーファイルが取り込まれる。
- プリプロセッサの実行結果に対して、字句解析と構文解析が行われる。
- 続いて、意味解析・最適化が行われて、アセンブリ言語のコードを作成する。
- アセンブラが実行可能形式のコードへ変換する。
- リンカが、スタートアップルーチンやライブラリを付加させて実行モジュールを作成する。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-3. シェルスクリプトによる定形処理	
対応する コースウェア	第1回（ログイン手順とコンパイル手順） 第2回（shell プログラミング）	

I-8-3. シェルスクリプトによる定形処理

シェルの種類やシェルスクリプトの概念、対話的シェルの操作方法を説明する。スクリプトを作成してから実行するまでの手順を示し、また基本的なシェルコマンドやシェルスクリプトの文法、記述方式について言及する。

【学習の要点】

＊ シェルスクリプトとは複雑な繰り返し処理や条件分岐を使って、複数のコマンドを逐次実行できるようにした簡易プログラムである。

＊ シェルスクリプトは、1行目に shebang を記述、2行目以降にプログラム本体を記述したテキストファイルとして作成する。

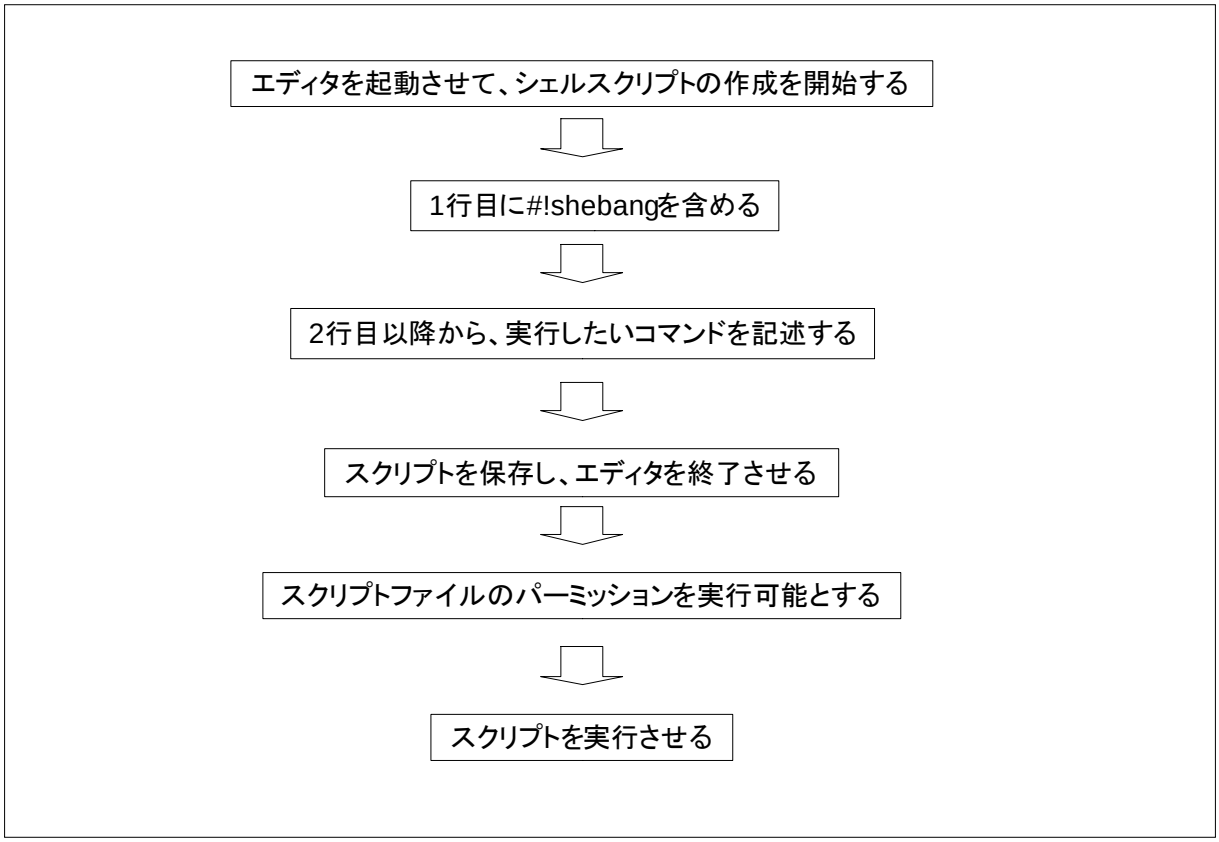

図 I-8-3.　シェルスクリプトの作成

【解説】

1) シェルスクリプトとは

複数のコマンドをまとめて実行することで、繰り返しの多い処理や一連の処理を簡単に実行できるようにした簡易プログラム。オペレータの指示をカーネルに伝えるためのシェルが、直接解釈し実行することができる。特徴は以下の通り。

* シェルごとに独自の文法が採用されている。
* 複雑な繰り返し処理や条件分岐などに対応している。
* コマンドを、そのままプログラムとして記述できる。
* 対話的なスクリプトを作成することも可能である。
* 引数を指定することで挙動が代わるスクリプトも作成することが可能である。

2) スクリプトの作成から実行まで

シェルスクリプトは、vi エディタなどを使ってコマンドなどをまとめたテキストファイルとして作成する。

* 1行目には shebang を含める。
  これはこのスクリプトの実行時に、どのインタプリタを使用するかを OS に伝えるものである。
  - Linux の標準シェルである bash の場合は、「#!/bin/bash」と明記する。
* 2行目以降に、実行させたいコマンドを並べて記述する。
  - コメント行は先頭に「 # 」を記述する。
* スクリプトファイルのパーミッションを、実行可能に変更する。
* パスを指定したディレクトリに配置するか、パスを指定しながらスクリプトを実行させる。
  - カレントディレクトリにある場合は、「./(スクリプトファイル名)」と指定する。

3) シェルスクリプトの構文

以下に基本的なコマンドの文法を明記する。

* 標準出力への表示
  - printf 文: 標準出力に対して、変数の値や文字列などをまとめて表示する。
    例) printf ″number is %d″ $number
  - echo 文: 標準出力に対して、文字列を表示する。
    例) echo ″Hello World″
* 標準入力からの入力
  - read 文: 入力された文字列を指定した変数に設定する。
    例) read NUMBER
* 繰り返し処理
  - for ループ: 変数に指定範囲内の数値を割り当てて、繰り返す。
    例) for number in $(seq 1 10)
* 判断文
  - if 文: 変数の持つ値によって処理を変更する。
    例) if [number != 0]; then 命令 fi

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-4. 低水準ファイル処理によるファイル操作	
対応する コースウェア	第3回（ファイル入出力プログラミング）	

I-8-4. 低水準ファイル処理によるファイル操作

低水準ファイルアクセスを行うシステムコールを使ってファイルを操作する方法を示す。ファイル操作のモード、ファイルの生成、オープンとクローズ、ファイルの読み書きとファイルパーミッションの変更、その他ファイル操作に必要なシステムコール郡について説明する。

【学習の要点】

＊ ファイルのアクセスには、高水準ファイルアクセスと低水準ファイルアクセスがある。

＊ 低水準ファイルアクセスを行うシステムコールを呼び出すために、様々な関数が用意されている。

＊ 低水準入出力関数は OS 依存の関数で、最低限の機能のみが用意されている。

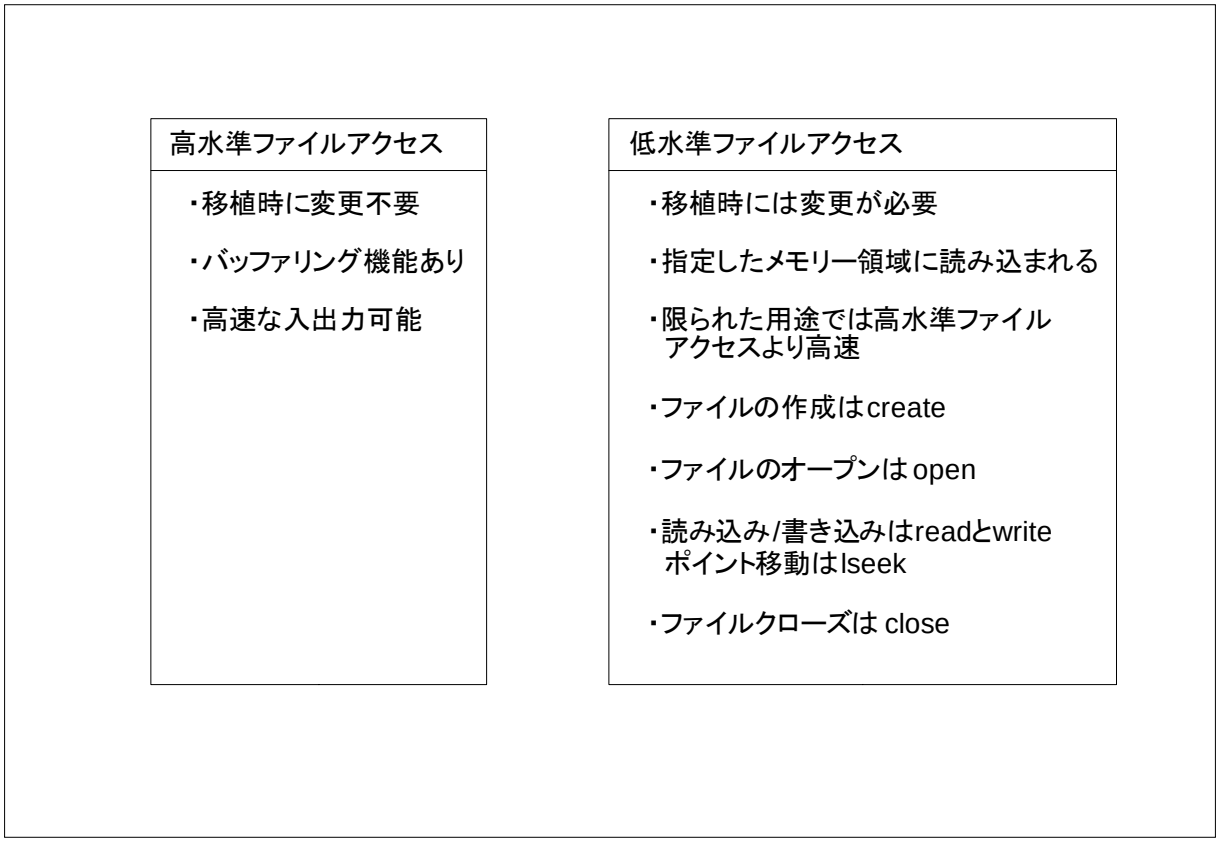

図 I-8-4.　低水準ファイルアクセス

【解説】

1) 低水準入出力関数の特徴

オペレーティングシステムに依存する関数で、かつ最低限の機能のみが用意されている。特徴を以下に示す。

* プログラムの変更が必要なため、簡単に他の OS へ移植させることができない。
* ファイルからの読込を行うと、プログラムが指定した領域に直接書き込まれる。
* 1バイトの読み込みであっても、最小単位である 1024 バイトをメモリ上に蓄える。ただし、次の1バイトの読み込みでは、メモリ上から読み込むため、高速な処理が可能。

2) 低水準ファイルアクセス入出力関数

以下に、低水準ファイルアクセス入出力関数を紹介する。関数の戻り値が-1 であった場合は、エラーが発生したことを意味する。

* creat 関数:ファイル作成を作成する。戻り値はファイルディスクリプタ。
  - 引数は、作成したいファイル名へのポインタと作成時のモード。
  - 作成時に指定するモードには、読み取り専用、読込書込みともに可能などが指定可。
* open 関数:ファイルをオープンする。戻り値はファイルディスクリプタ。
  - 引数は、作成したいファイル名へのポインタとオープン時のモード。
  - オープン時に指定するモードには、読込専用、書込専用、読込書込み可などが指定可。
* read 関数:ファイルからの読込み。戻り値は読み込んだサイズ。
  - 引数は、ファイルディスクリプタ、読み込みたいバイト数、格納するバッファ。
  - ファイルディスクリプタは、creat 関数や open 関数からの戻り値。
* write 関数:ファイルへの書き込み。戻り値は実際に書き込んだサイズ。
  - 引数は、ファイルディスクリプタ、書き込みたい情報、書き込みたいサイズ。
* lseek 関数:入出力を行いたい位置の指定。戻り値はファイル先頭からのバイト数。
  - 引数は、ファイルディスクリプタ、移動したいバイス数、開始したい位置。
  - 第三引数には、L_SET(先頭から)、L_XTND(終端から)などの指定が可能。
* close 関数:ファイルクローズを閉じる。0が戻るとクローズ成功を意味する。
  - 引数は、ファイルディスクリプタを指定する。

3) その他の低水準ファイル処理関数

その他の、低水準ファイル処理関数を紹介する。

* chmod 関数、fchmod 関数; ファイルのモードを変更する。
* chown 関数、fchown 関数、lchown 関数: ユーザ ID、グループ ID を変更する。
* umask 関数: プロセスによりファイル作成時のマスク値を設定、変更する。
* access 関数: ファイル操作のアクセス権を検査する。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-5. 標準入出力ライブラリによるファイル操作	
対応する コースウェア	第3回（ファイル入出力プログラミング）	

I-8-5. 標準入出力ライブラリによるファイル操作

標準入出力ライブラリを用いてファイルを操作する方法を示す。低水準ファイル処理のそれぞれに対応する関数を紹介し、さらに書式付き入出力や利用上の留意点も示す。また一時ファイルを作成する関数についても触れる。

【学習の要点】

* 低水準関数には最低限の機能しか用意されていないが、高水準関数であれば便利な機能も用意されている。
* 高水準関数には、ファイル操作のための標準入出力ライブラリなどが用意されている。
* 高水準関数を使用した場合であっても、基本的なファイル操作の流れは同じである。

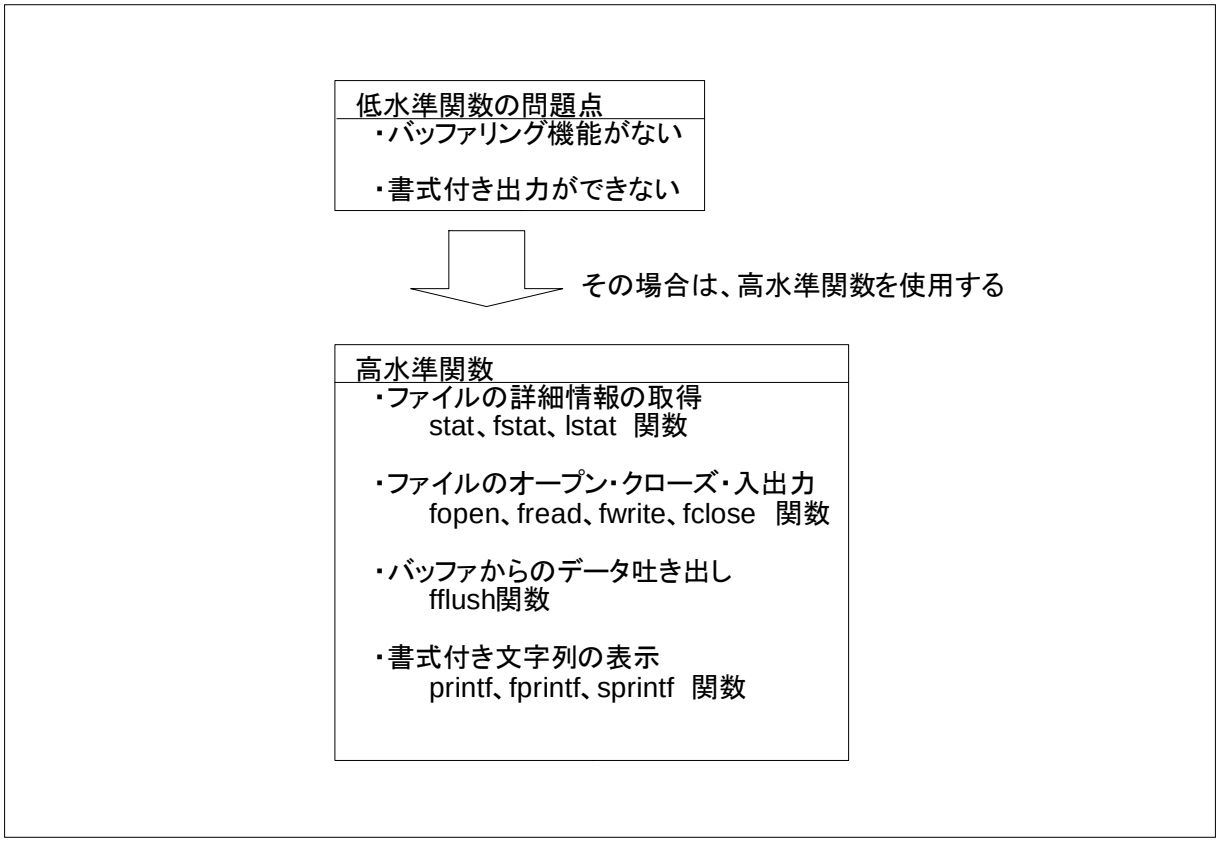

図 I-8-5.　高水準関数

【解説】

4) ファイル操作に関して

低水準関数には最低限の機能しか用意されていないため、不便な面も多い。それと比べて高水準関数であれば、多少は便利な機能も用意されている。例えば、ファイルの詳細情報(i ノード・ファイルモード・UID・GID・時間情報など)を収集したい場合には、以下に示す関数などを使用するとよい。

* stat、fstat、lstat 関数
  個々の関数の違いは、情報を得たいファイルの指定方法である。この関数が正常に終了すると、引数で指定した stat 構造体に、ファイルの情報が書き込まれる。
* stat 構造体
  i ノードやファイルモード、UID と GID、さらには時間情報など、指定したファイルが持つ多くの情報を書き込みための領域。

5) 標準入出力ライブラリに関して

高水準関数を使用した場合であっても、はじめにファイルをオープンして読み込みや書き込みを行い、最後にクローズする流れは同じである。

* ファイルのオープン・クローズや入出力を行う場合には、以下の関数を使用する。
  fopen, fread, fwrite, fclose
* バッファに格納されているデータを吐き出す場合には、以下の関数を使用する。
  fflush
* ファイル位置子を移動させる場合には、以下の関数を使用する。
  fseek
* 入出力ストリームや標準入出力から、1文字読み込んだり書き込んだりする場合には、以下の関数を使用する。
  fgetc, getc, getchar, fputc, putc, putchar

6) 書式付き入出力に関して

低水準関数には書式付き出力などの機能がない。書式を考慮した入出力を行いたい場合には、以下の関数を使用する。

* 書式付き文字列の表示関数
  printf、fprintf、sprintf
* 書式付き入力関数
  scanf、fscanf、sscanf

7) 一時ファイルの作成に関して

* 既存のファイルと重複しない新たなファイルの名前を作成する。
  tmpnam
* プログラム終了時に自動的に削除される一時的なバイナリファイルを作る。
  tmpfile

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-6. ファイルシステムの操作	
対応する コースウェア	第4回（ファイルシステム）	

I-8-6. ファイルシステムの操作

ファイルシステムの概念や構造、ファイルシステムの種類と各ファイルシステムの特徴について説明し、複数のディスクブロックをひとつのツリーとして扱う考え方とその方法を解説する。またディレクトリをプログラムから操作する方法や特殊なディレクトリについて説明する。

【学習の要点】

* ファイルシステムとは、コンピュータ内にある資源の操作や保護を実現する仕組みである。
* ファイルシステムは、木構造となっている。
* ファイルシステムの種類としては、ext2, ext3 などが存在する。

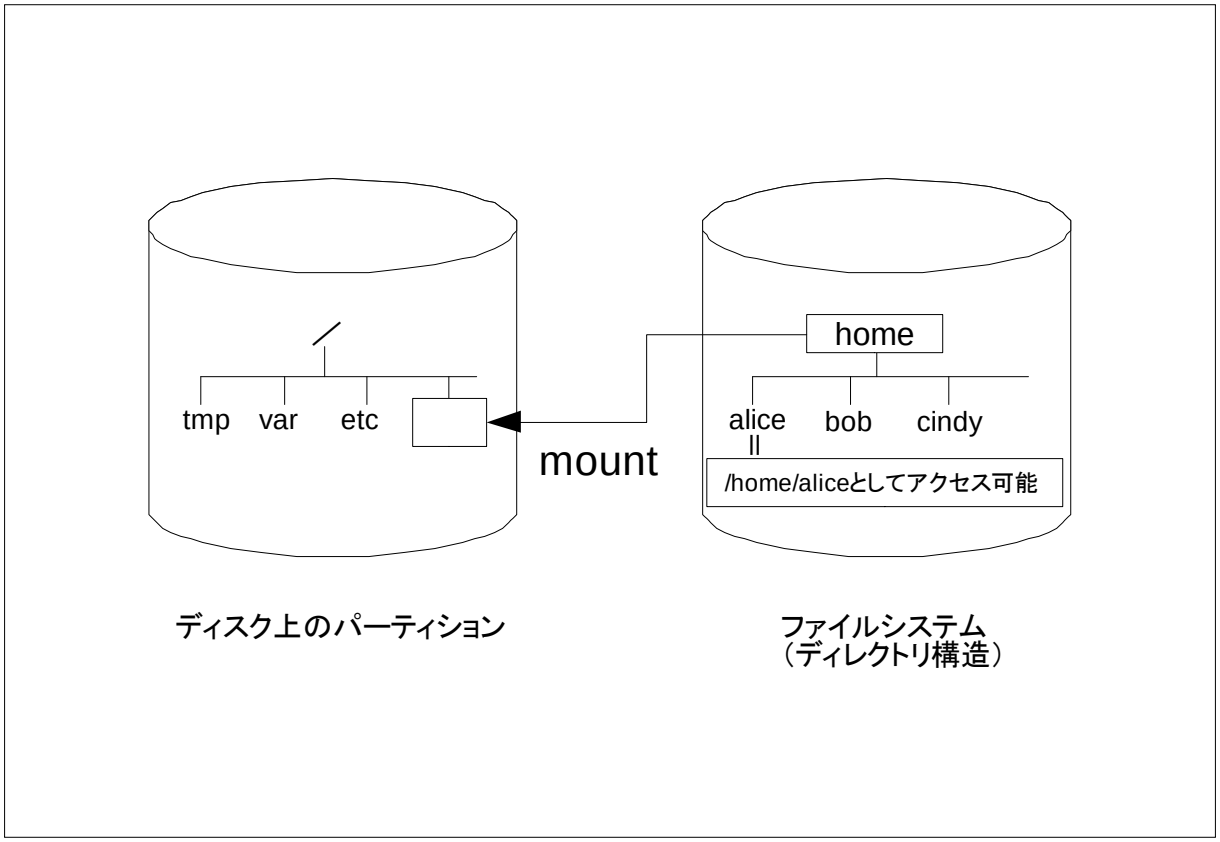

図 I-8-6.　ファイルシステムの操作

【解説】

1) ファイルシステムの概念と構造

ファイルシステムは、ファイルやディレクトリなどコンピュータ内部にある資源を操作したり保護したりするための仕組みである。

* ファイルシステムの中に存在するもの。
  データを保持するためのファイル、ディレクトリファイル、スペシャルファイルなどがある。
* ファイルシステムに関する情報の収集方法。
  statfs 関数により、スーパーブロック等の情報取得が可能である。
* ファイルシステムの使用方法。
  ディスク上のパーティションを任意のディレクトリに mount・umount することで、ファイルシステムへの追加や削除が可能となる。
  mount するディレクトリのことをマウントポイントと呼ぶ。全てのマウントポイントの情報は、/etc/fstab ファイル内に格納される。
* ファイルシステムの構造
  ファイルシステムは、逆さまの木構造となっている。最上位には"/(ルート)"ディレクトリが存在し、その下には木の枝が伸びるように様々なディレクトリが用意されている。
  ディスク内のパーティションがマウント済みの状態である場合、親のディレクトリや配下のディレクトリ(子供のディレクトリ、サブディレクトリなどと呼ぶ)は異なるパーティションにある可能性もある。
  ファイルシステムにより、データが複数のパーティションに分割されていることをユーザが意識せずに、操作を行うことが可能となる。
* ファイルシステムの種類
  FAT, ext2, ext3, ReiserFS, NTFS などが存在する。
* ファイルの管理方法
  ext2／ext3 ファイルシステムでは、ファイルを管理する場合にiノードと呼ばれる機能を利用している。iノードには個々のファイルの詳細情報が収められており、iノードテーブルと呼ばれる領域に納められている。個々のファイルにはiノード番号と呼ばれる情報があり、iノードテーブル内の場所を示している。

2) プログラムからの操作

プログラムから、個々のディレクトリへアクセスを行う場合には、以下に示す関数を使用する。

* ディレクトリのオープンとクローズ
  opendir, closedir
* ディレクトリの読み込みと関連する関数
  readdir, telldir, seekdir

3) 特殊ディレクトリ

ファイルシステムの中には、/proc という名称の特殊ディレクトリが存在する。

* /proc ディレクトリは、実際にディスク上に作成されるディレクトリやファイルではなく、カーネルの認識している情報をファイルシステムのインタフェースを介して参照するための仮想ファイルシステムである。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-7. コマンドラインオプションと環境変数のプログラムへの渡し方	
対応する コースウェア	第5回（UNIX 環境）	

I-8-7. コマンドラインオプションと環境変数のプログラムへの渡し方

シェルからプログラムへパラメータを渡す方法として、コマンドライン引数による方法と環境変数を用いる方法を説明する。渡されたパラメータをC言語プログラム内部で処理する方法やプログラムからの返り値の取扱いについて解説し、代表的な環境変数についても説明する。

【学習の要点】

＊ プログラムは起動時に指定されたパラメータの値により、その振る舞いが決定される。

＊ パラメータは、コマンドライン引数としても指定することが可能である。

＊ またパラメータは、環境変数として指定することも可能である。

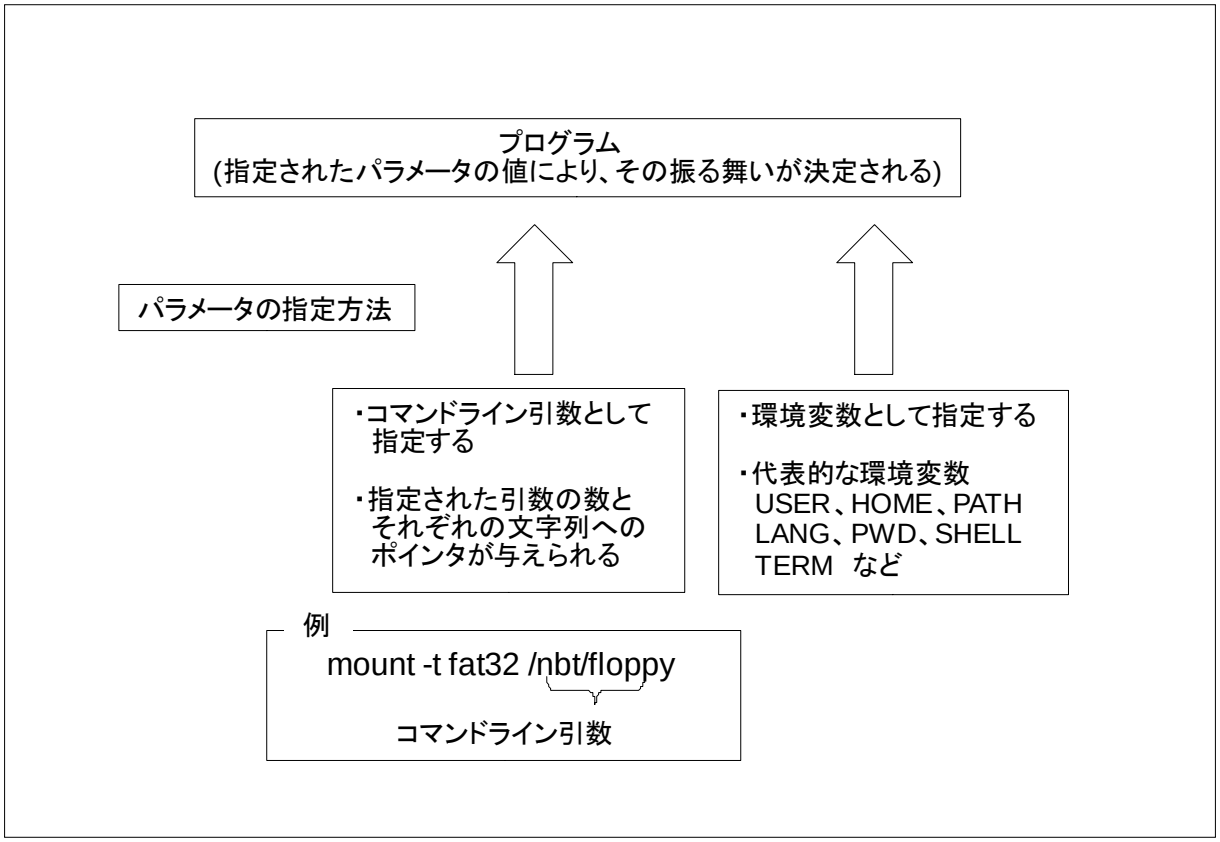

図 I-8-7.　プログラムへのパラメータ

【解説】

1) プログラムへのパラメータ

プログラムは起動時に指定されたパラメータの値により、その振る舞いが決定される。個々のプログラムに対してパラメータを指定するには、以下の方法がある。

＊ コマンドライン引数として指定する。
コマンドライン引数として与えられた文字列は、空白を区切り文字として複数の文字列に分けられ、指定された引数の数とそれぞれの文字列へのポインタが、プログラムに与えられる。

＊ 環境変数として指定する。
事前に、パラメータ値を環境変数に設定しておけば、以下の関数で参照が可能となる。

- printenv コマンド ： 環境変数の表示を行う。
- set コマンド ： 環境変数を設定する。
- getenv 取得関数 ： 環境変数を検索し、この変数の示す値を返す。
- setenv 設定関数 ： 環境変数を作成する。
- putenv 変更関数 ： 環境変数の値を更新する。
- unsetenv 削除関数 ： 指定した環境変数を削除する。

2) C言語プログラムにおけるパラメータ処理

C言語で記述された main 関数に対してもパラメータの指定が可能であり、これらは main 関数の引数として参照することが可能である。

＊ コマンドライン引数の処理方法
main 関数への引数を受け取るためには、以下のように指定すればよい。

- main(int argc, char *argv[]) {
  argc がパラメータの個数を示し、argv に指定されたパラメータが順次納められている。

＊ 環境変数として指定されたパラメータの処理方法
getenv 関数を用いて環境変数の示す値を受け取るには、getenv 関数への引数として環境変数の名称を指定する。戻り値は該当する変数の持つ値へのポインタとなる。

＊ 各関数からの戻り値
各関数からは、必ず何らかの戻り値が存在する。戻り値を参照するには環境変数を以下の方法で参照すればよい。 例） # echo $?

3) 代表的な環境変数

Linux が使用している代表的な環境変数には、以下に挙げる変数がある。

- USER ： 現在の使用者名（ログイン名）が設定されている。
- HOME ： 現在の使用者のホームディレクトリ名が設定されている。
- PATH ： 使用可能なコマンドが置かれているディレクトリ名が設定されている。
- LANG ： 現在使用中の言語が設定されている。
- PWD ： 現在のカレントディレクトリ名が設定されている。
- SHELL ： 現在使用中のシェルの名称が設定されている。
- TERM ： 標準で使用するターミナルの名称が設定されている。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-8. ライブラリとプログラムの関係	
対応する コースウェア	第5回（UNIX 環境）	

I-8-8. ライブラリとプログラムの関係

プログラムをコンパイルしてできあがる実行ファイルと、実行ファイルから関数呼び出しで利用するライブラリの関係を説明する。また共有ライブラリの考え方と、実行時の動作、ldd コマンドによる利用ライブラリの一覧や size コマンドによるメモリ利用状況の把握などを解説する。

【学習の要点】

＊ ライブラリとは、複数のプログラムが共通して利用するコードをまとめたファイルである。

＊ ライブラリ自体は単独で実行することはできない。

＊ 実行プログラムがライブラリをリンクする方法には、動的リンクと静的リンクとが存在する。

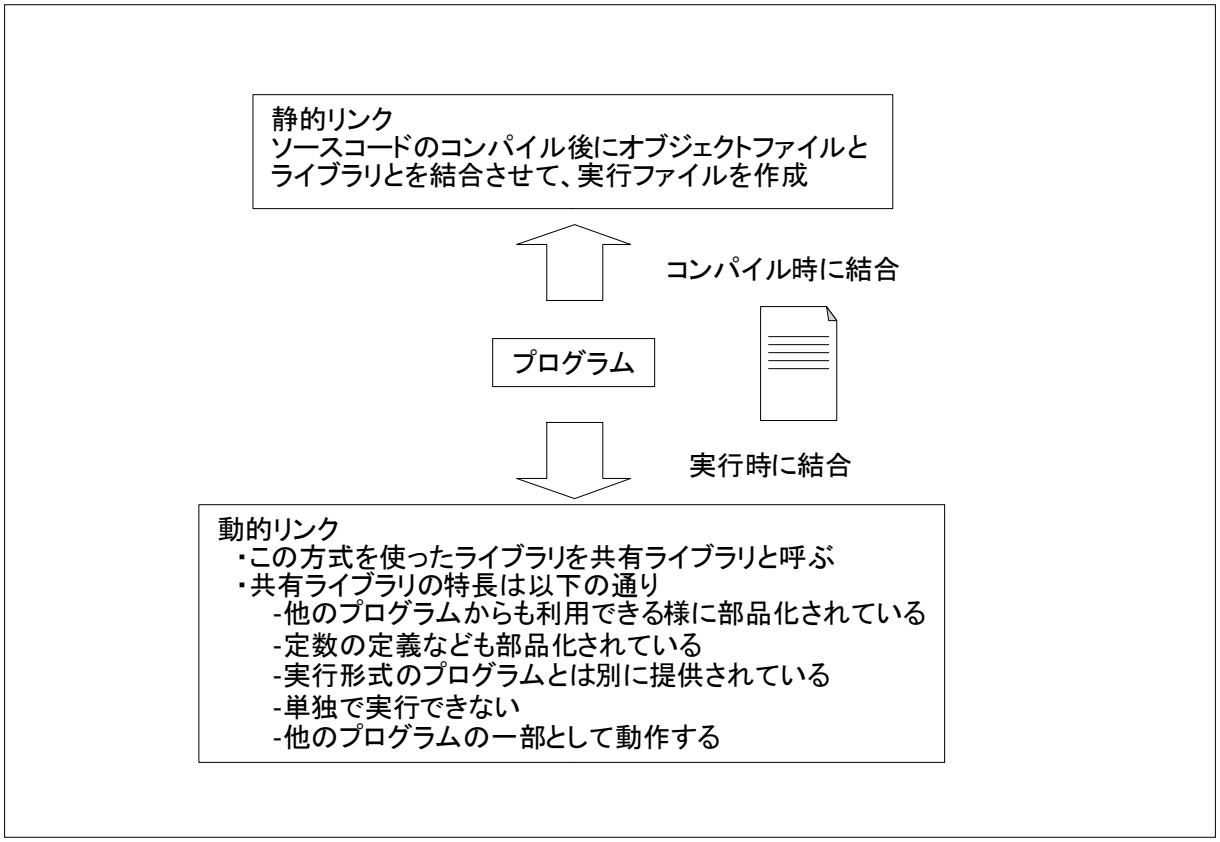

図 I-8-8.　実行ファイルとライブラリの関係

【解説】

1) ライブラリとは

ライブラリとは、多くのプログラムが作成されている場合に、その中のどのプログラムにとっても共通な処理として必要と考えられる機能（プログラム）をまとめたもの。

＊ 様々なプログラムから利用できるように部品化されている。

＊ プログラムに限定されることはない。例えば定数の定義なども部品化されている。

＊ 単独のファイルとして、実行形式のプログラムとは別に提供されている。

＊ ライブラリ自体を単独で実行することはできない。

＊ 基本的に他のプログラムの一部として動作する。

＊ 複数のプログラマが、同じ目的や機能を持つプログラムを、個々に開発してしまう無駄を省く。

2) 実行ファイルとライブラリの関係

実行プログラムがライブラリをコールする方法には、動的リンクと呼ばれる方法と静的リンクと呼ばれる方法が存在する。

＊ 動的リンク
動的リンクとは、プログラムの起動時もしくは起動後のプログラム実行中に初めてライブラリを結合される方式である。共有ライブラリは動的リンクの仕組みを利用して実現される。

＊ 静的リンク

＊ 静的リンクとは、ソースコードのコンパイル後にオブジェクトファイルとライブラリとを結合させて、実行ファイルを作成する方法である。

＊ Linux の場合、/lib、/usr/lib、/usr/local/lib などのフォルダに置かれる libfoo.a や libfoo.so といったファイルがライブラリである。ちなみに、ファイル名は常に lib という文字列で始まり、.so で終わる。静的ライブラリの場合は.a で終わる。

3) 共有ライブラリとは

様々なプログラムが同じライブラリを利用する場合、それぞれがメモリ上にコードを展開するとメモリの使用量が膨れ上がってしまう。それを避けるために、同一のコードを共有して利用できるように工夫されたライブラリのことを、共有ライブラリという。

4) ライブラリ関連コマンド

個々の実行ファイルとライブラリの関係などは、以下に示すコマンドなどにより参照可能である。

＊ 利用ライブラリの一覧表示
ldd コマンド

＊ プログラムの使用メモリの状況
size コマンド

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-9. ライブラリの種類とライブラリの利用方法	
対応する コースウェア	第6回 (ライブラリの利用方法と作成手順)	

I-8-9. ライブラリの種類とライブラリの利用方法

ライブラリには static ライブラリと shared ライブラリの2種類があることを説明し、それぞれの利点と特徴、利用方法について解説する。また、ライブラリを利用したシステムプログラミングの例を示す。ライブラリで提供される関数とシステムコールの違いについての説明も行う。

【学習の要点】

* 静的ライブラリとは、プログラム(実行形式)の構築時に組み込まれるライブラリである。
* 共有ライブラリとは、プログラム実行と同時にメモリ上に展開されるライブラリである。
* 動的ライブラリとは、プログラム実行中の任意の時点で読み込まれるライブラリである。

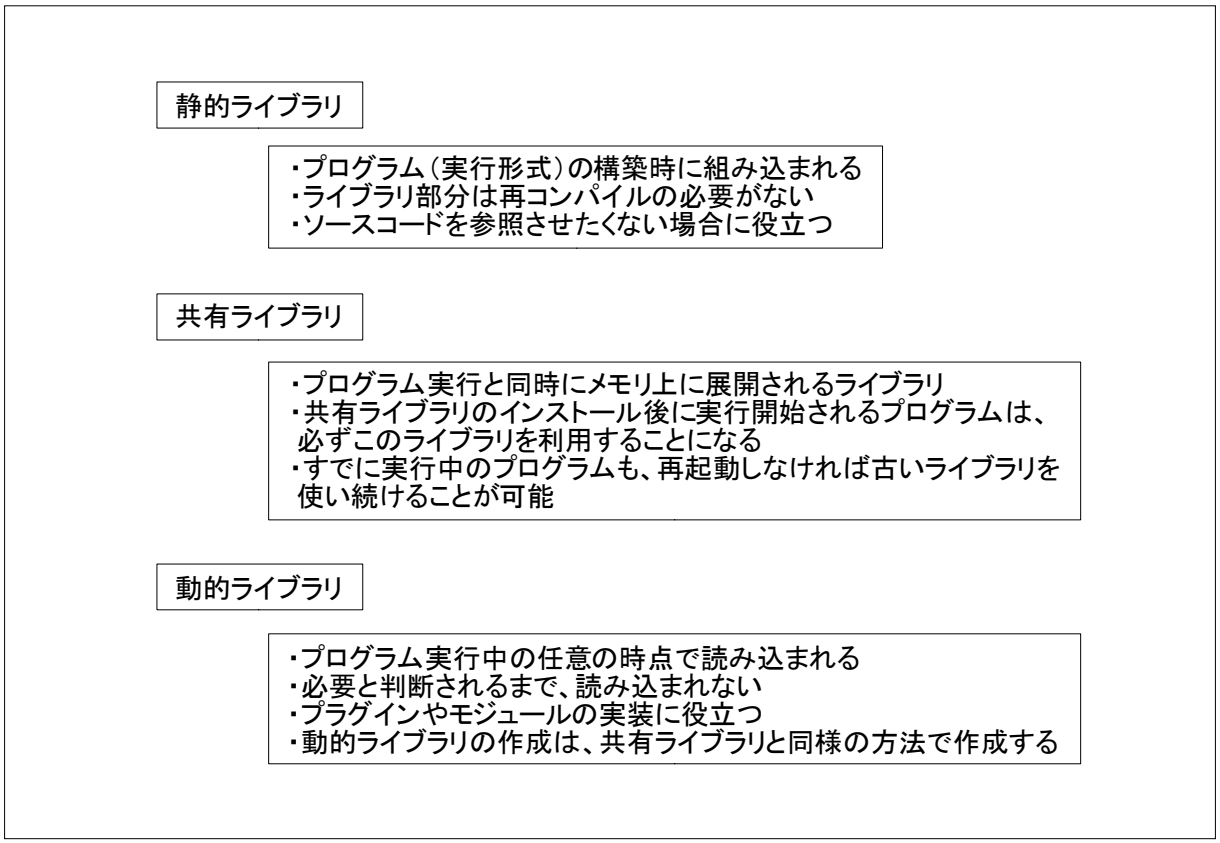

図 I-8-9.　ライブラリの種類

【解説】

1) ライブラリの種類

ライブラリとは、実行させることが可能な複数個のプログラムに対して、そのプログラムにとっても共通であると判断できる部分のプログラムを集めたファイルである。ライブラリには、静的ライブラリと共有ライブラリがある。共有ライブラリには、動的ライブラリも含まれている。

＊ 静的ライブラリ（static library）
プログラム（実行形式）の構築時に組み込まれるライブラリ。
- ライブラリ部分は再コンパイルの必要がない。
- ライブラリとして提供するプログラムの、ソースコードを参照させたくない場合に役立つ。

＊ 共有ライブラリ（shared library）
プログラム実行と同時にメモリ上に展開されるライブラリ。
- 共有ライブラリのインストール後に実行開始されるプログラムは、必ずこのライブラリを利用することになる。
- すでに実行中のプログラムも、再起動しなければ古いライブラリを使い続けることが可能。

＊ 動的ライブラリ(dynamically loaded library)
プログラム実行中の任意の時点で読み込まれるライブラリ。
- 必要と判断されるまで、読み込まれない。
- プラグインやモジュールの実装に役立つ。
- 動的ライブラリの作成は、共有ライブラリと同様の方法で作成する。

2) ライブラリの利用方法

個々のライブラリを使用する際には、ライブラリ毎に異なる特徴を持っているので注意する。

＊ 静的ライブラリ
実行可能プログラムを作成するタイミングで、静的ライブラリを取り込む。コンパイルの際に gcc コンパイラを使用しているのであれば、ライブラリの指定に-l オプションを指定する。

＊ 共有ライブラリ
/usr/lib などのディレクトリにファイルを配置して、ldconfig コマンドを実行する。実行可能プログラムをコンパイルする際に、-L オプションで共有ライブラリを指定する。

＊ 動的ライブラリ
ライブラリをオープンし、シンボルを検索し、エラーを処理し、ライブラリを閉じる、という関数が存在する。それらを呼び出すことによって動的ライブラリを利用することができる。

3) ライブラリとシステムコール

＊ ライブラリ
基本的に OS には依存せず、かつ複数のアプリケーションプログラムの中で共通と判断される処理を提供する。内部的にシステムコールを呼び出しているものも存在する。

＊ システムコール
OS 依存の関数である。また必要最低限なものしか用意されていない。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	8　Linux のシステムプログラミングに関する知識 I	基本
習得ポイント	I-8-10. メモリ管理、ファイル管理とデータベースの利用	
対応する コースウェア	第7回（データの管理）	

I-8-10. メモリ管理、ファイル管理とデータベースの利用

OSが管理するメモリの動的な利用方法、ファイルの排他処理として行うロックの方法、Berkeleyデータベースの利用など、システムプログラムから大規模なデータを管理する方法について解説する。

【学習の要点】

* 多くのデータ構造では、処理の都合で動的にメモリを確保する必要がある。
* ファイルのロックは、マルチプロセス環境においてデータの安全性確保のために重要な操作である。
* データベースを使用すると、ファイルのロックを気にせずにデータのアクセスが可能となる。

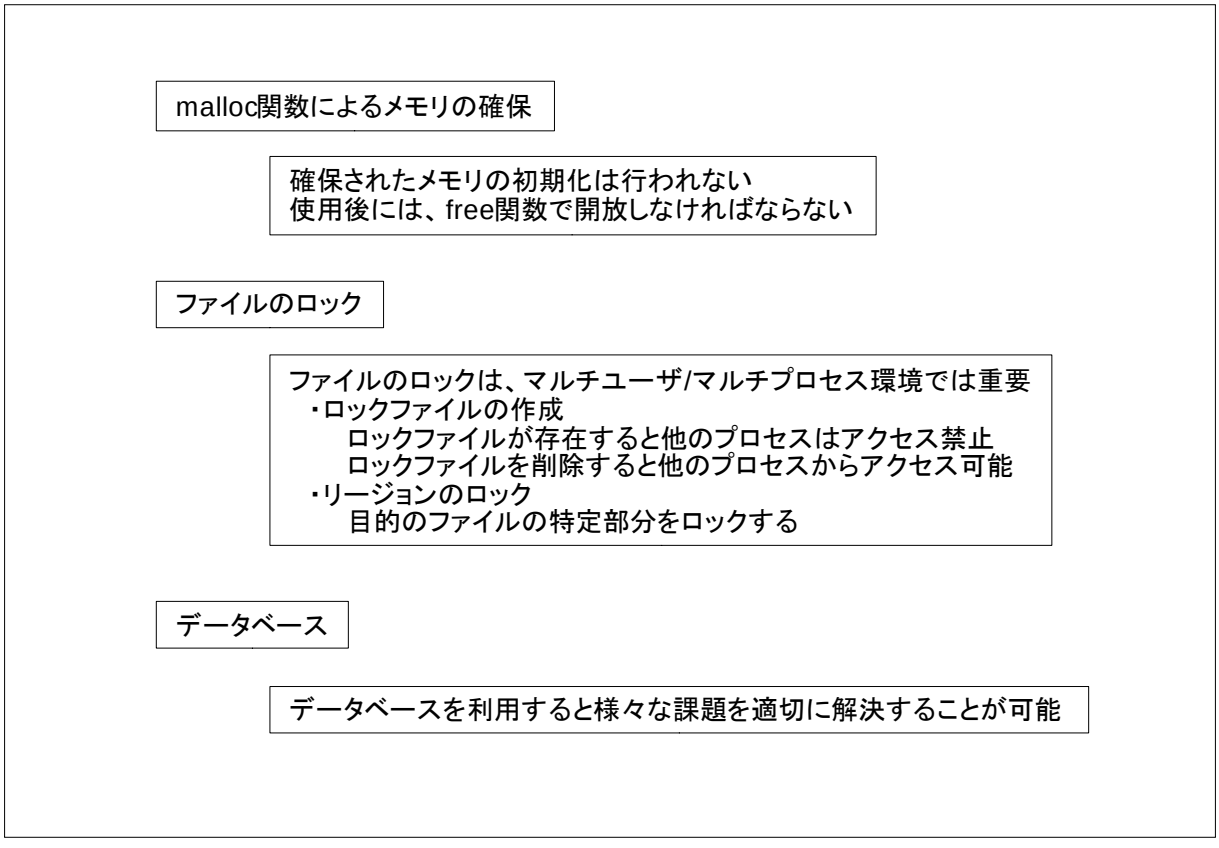

図 I-8-10.　メモリ管理とファイル管理

【解説】

1) メモリの管理

アプリケーションプログラムで確保可能なメモリ領域は、事前に決まっている。処理の都合で定められたサイズ以上の領域が必要な場合は、malloc 関数を利用してメモリの確保を行う。

＊ 確保されたブロックは、初期化されない。

＊ 使用後には必ず free 関数などで解放する。

2) ファイルのロック

マルチプロセス環境では、複数のプロセスから1つのファイルへのアクセスを集中させることも可能であるため、データの安全性確保のためにロックファイルの作成が重要となる。

＊ ロックファイルの作成
使用するリソースのロックファイルを作成すると、他プロセスからはアクセスできなくなる

- ロックファイルの作成に、open 関数を利用する。引数には O_CREAT と O_EXCL を指定。
- ロックファイルが新規に作成されれば、そのリソースの専有が可能となる。
- ロックファイルが存在すれば、エラーとなるため、リソースへのアクセスは不可能となる。

＊ リージョンのロック
目的とするリソースの特定の部分のみをロックする。別プロセスからはロックされていない部分へのアクセスが可能。

- リージョンのロックは、fcntl 関数、もしくは lockf 関数を使う。以下は fcntl 関数の引数。
- F_GETLK 引数:オープン済みリソースのロック情報を取得。
- F_SETLK 引数:オープン済みリソースの一部をロックする。もしくは解除する。
- F_SETLKW 引数:ロックできるまで待機する。
- 実際の読み書きは read 関数と write 関数を使用する。

3) データベース

データベースを利用すれば、ファイルのロックやリージョンのロックなどは気にせずに、データのアクセスが可能となる。さらに、サイズの異なるデータ書込みや索引機能を使ってのデータ管理も可能となる。

＊ dbm データベース
滅多に書き換えられることはないのだが、頻繁に読み込みが行われるデータなどの管理には、dbm データベースなどがソリューションとなる。dbm へのアクセスは、以下の関数を利用する。

- dbm_open ： データベースのオープン。または新規作成。クローズは dbm_close を使用。
- dbm_store ： データベースにデータを入力する。
- dbm_fetch ： データベースからデータを取得する。
- dbm_delete ： データベースからデータを削除する。
- dbm_firstkey と dbm_nextkey ： 一緒に設定する。全項目のキー走査を行う。

＊ その他のデータベースには、ndbm、gdbm などがある。