6. Linux カーネルに関する知識 II

1. 科目の概要

Linux オペレーティングシステムの中核をなすカーネルの動作について、メモリ管理、プロセス管理、といった基本動作の詳細と、カーネルによるファイルシステムやネットワークの取り扱いについて、具体的な動作原理を説明する。

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
II-6-1. アドレス空間	Linuxカーネルが取り扱うアドレス空間として、プロセス空間とカーネル空間という概念があること、物理的なメモリ領域に対するアドレス空間と仮想的なアドレス空間が存在し、それらは適宜マッピングされて管理されることの基本的な原理を解説する。	6
II-6-2. アドレス変換(ページ管理)	物理メモリに対するアドレス空間と仮想メモリに対するアドレス空間のマッピングとして用いられている「ページ」の概念を解説する。ページ変換テーブル、バディシステムといったページ管理に用いられる基本的な技術を紹介し、ページ管理の効果を説明する。	6,7
II-6-3. メモリの割り当て方法	メモリ管理を理解する一環として、動的なメモリ割り当て方式の原理を説明する。動的メモリ割り当てに必要な要素技術を説明するとともにメモリ割り当てのアルゴリズムを紹介し、いくつかあるアルゴリズムの得失を比較する。	7
II-6-4. プロセス空間の管理方法	プロセス空間を管理する方法の具体的な技術について説明する。スワップの概念を示し、ページキャッシュ、ページフォルト、デマンドページングといった個々の概念を解説する。また関連する話題として実メモリが不足してきた際に処理が必要となるガベージコレクション等の技術についても述べる。	8
II-6-5. 仮想ファイルシステム	ファイルシステムの違いを意識することなくあらゆるファイルシステムへ一元的にアクセスできるようにするための仮想ファイルシステムについて解説する。仮想ファイルシステムの概念と特徴を説明し、実際に操作がどのように行われるかについて示す。	9
II-6-6. カーネルによるファイル操作	ファイルシステムに対する一般的な操作を説明する。ファイルのオープンとクローズ、データの読み書きなど基本的な操作から、ディスクへの書き出しがどのように行われるか、同期がとられるタイミングといったカーネルの内部動作についても解説する。	10
II-6-7. 特別なファイル	ブロックデバイスやパイプ、procファイルシステムのような疑似ファイルシステムなど、Linuxカーネルからファイルシステムインタフェースによりアクセスすることができる特別なファイルを説明し、その意義とメリットについて触れる。	11
II-6-8. プロトコルスタック	OSIの参照モデルについて説明し、TCP/IPとの対応関係を示す。また、Linuxのソケットインタフェースからどのようにネットワークが取り扱われるか、Linuxカーネルとネットワークのインタフェースについての説明を加える。	12
II-6-9. IPネットワークの基本	IPネットワークの基本的な概念と、IPネットワークで利用される基本的な要素技術について解説する。IPパケットの構成やルーティングの概念、送受信がどのように行われるかといった基本的な技術について説明する。	13
II-6-10. UDPとTCP	インターネットで広く利用されているUDPによる通信方式と、信頼性を高めたTCPによる通信方式について説明する。またとくにTCPに関しては通信手順の詳細を示し、さらにフロー制御や輻輳制御など高度な通信制御技術も示す。	14,15

【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

「6. Linux カーネルに関する知識Ⅱ」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
6. Linuxのカーネルに関するスキル	<Linuxカーネル概論>	<スケジューリング>	<割り込みと遅延>	<システムコール>	<プロセス管理>	<メモリ管理(1)>	<メモリ管理(2)>	<メモリ管理(3)>	<ファイル管理(1):仮想ファイルシステム>	<ファイル管理(2):ファイルの操作>	<ファイル管理(3):特殊ファイル>	<ネットワーク(1):ソケットインタフェース>	<ネットワーク(2):IPとUDP>	<ネットワーク(3):UDPとTCP>	<ネットワーク(4):TCPフロー制御と輻輳制御>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_06.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.データ割り当てと変換	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.ソフトウェアの標準化	
	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 [6-13,14]	IT-NET2.ルーティングとスイッチング [6-13]	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 [6-13,14,15]	CE-NWK3.LANとWAN [6-15]	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 [6-1,2] [6-6,7,8]	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 [6-2]	CE-OPS1.並行性 [6-1,2,4]	CE-OPS2.スケジューリングとディスパッチ [6-2,3,4,5]	CE-OPS3.メモリ管理 [6-3] [6-6,7,8]	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 [6-2,5] [6-9]	CE-OPS7.OSの概要	CE-OPS8.設計の原則 [6-1,3]	CE-OPS9.デバイス管理	CE-OPS10.システム性能評価		
コンピュータアーキテクチャ	13	CE-CA0 コンピュータのアーキテクチャと構成	CE-CA00.歴史と概要	CE-CA01.コンピュータアーキテクチャの基礎	CE-CA02.メモリシステムの構成とアーキテクチャ [6-6,7,8]	CE-CA03.インタフェースと通信	CE-CA04.デバイスサブシステム	CE-CA05.CPUアーキテクチャ	CE-CA06.性能・コスト評価	CE-CA07.分散・並列処理	CE-CA08.コンピュータによる計算	CE-CA09.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 カーネルの機能の具体的な実装とコマンドとが挙げられる。他の多くの項目は、一般的なオペレーティングシステムの機能について Linux を通して習得する。

科目名	第6回	第7回	第8回	第9回	第10回	第11回	第12回	第13回	第14回	第15回
6.Linux カーネルに関する知識Ⅱ	(1)プロセス空間とカーネル空間 (2)仮想アドレス空間 (3)ページ変換テーブル (4)TLB (Translation Lookaside (5)メモリキャッシュ	(1)実メモリの管理 (2)動的メモリ割り当て	(1)スワップ (2)デマンドページング (3)ページキャッシュ (4)ページフォルト (5)ガーベジコレクション	(1)VFS (Virtual File System) (2)inode、super block、directory (3)ファイル操作 (4)マウント (5)アンマウント	(1)ファイルの操作 (2)ファイルのリード／ライト (3)そのほかのファイル操作	(1)特殊ファイル (2)擬似ファイルシステム (3)ローカルファイルシステム	(1)OSI 参照モデルと TCP/IP (2)ソケットインタフェース (3)ネットデバイス	(1)IP	(1)UDP (2)TCP	(1)フロー制御 (2)TCP の受信と送信 (3)再送処理 (4)輻輳制御

（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	6　Linux カーネルに関する知識　II	応用
習得ポイント	II -6-1. アドレス空間	
対応する コースウェア	第 6 回　メモリ管理 (1)	

II -6-1. アドレス空間

Linux カーネルが取り扱うアドレス空間として、プロセス空間とカーネル空間という概念があること、物理的なメモリ領域に対するアドレス空間と仮想的なアドレス空間が存在し、それらは適宜マッピングされて管理されることの基本的な原理を解説する。

【学習の要点】

* アドレス空間とは、カーネルまたはプロセスが参照することのできるメモリ領域である。
* Linux には、仮想アドレス空間の概念があり、プロセス毎に物理アドレス空間とは独立して管理されている。プロセスは仮想アドレスを用いて仮想アドレス空間にアクセスするが、このとき、仮想アドレスから物理アドレスへのアドレス変換が行われる。
* 仮想アドレス空間の元では、プロセスは二次記憶などの主記憶以外のメモリシステムに存在することができ、物理アドレス空間よりも広い空間を使用することができる。
* 仮想アドレス空間は、カーネル用に予約されたカーネル空間と、ユーザプロセスが利用可能なプロセス空間とに分けられる。

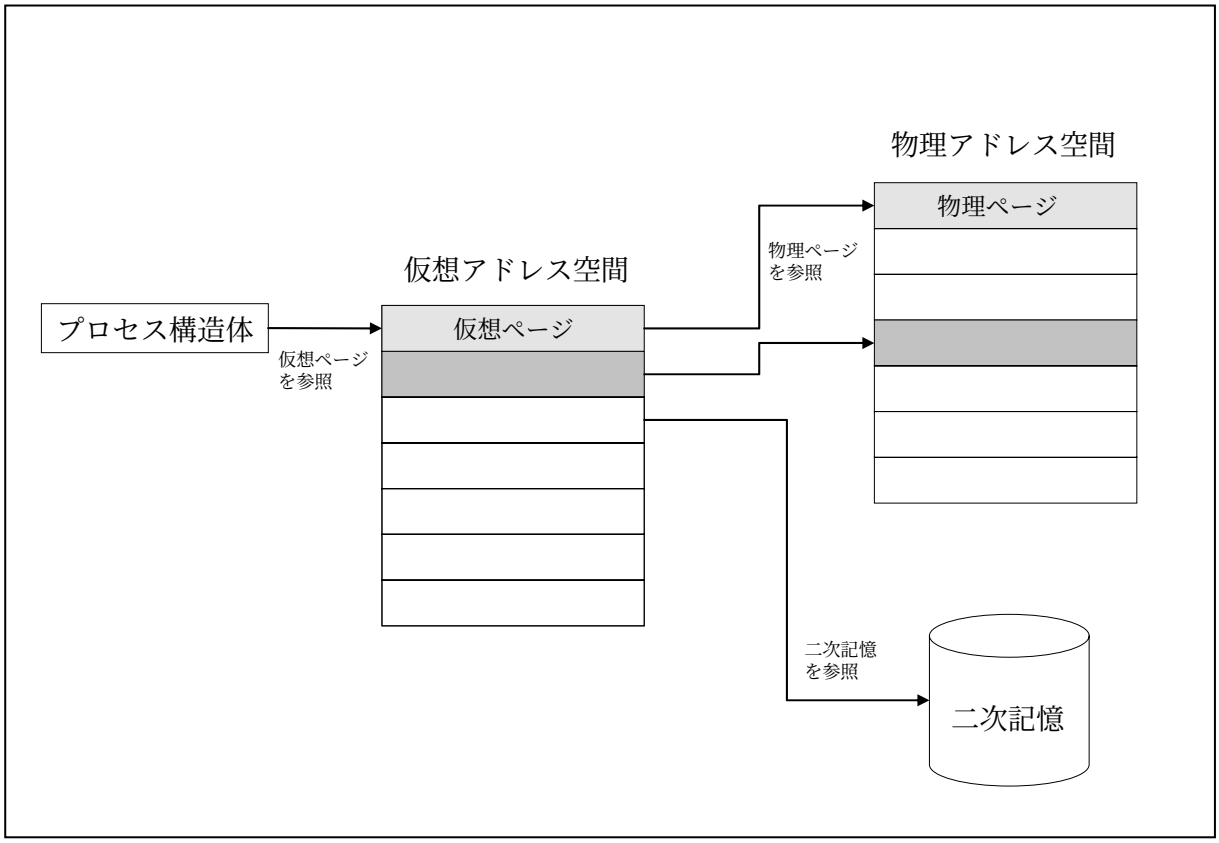

図 II-6-1.　アドレス空間

【解説】

1) アドレス空間とは

* システムが認識しているメモリ領域のことで、「メモリアドレス空間」とも呼ばれる。
* アドレス空間は、プロセスを実行するにあたって、そのプロセス状態を格納する場所として使用される。プロセスが終了するとアドレス空間からプロセス状態が削除される。
* アドレス空間には、「物理アドレス空間」と「仮想アドレス空間」とがある。
  - 物理アドレス空間

    CPU が直接アクセスすることのできる「主記憶（メインメモリ）」のもつアドレス空間。
  - 仮想アドレス空間

    システムが物理アドレス空間とは独立して仮想的に管理するアドレス空間。
* Linux では、プロセスが起動されると、そのプロセス状態は仮想アドレス空間上に置かれる。

2) 仮想記憶システム

* 仮想アドレス空間は、下記の特徴を持つ。
  - 仮想アドレス空間は、その領域すべてが主記憶上に存在している必要がない。
  - 仮想アドレスは二次記憶上の場所を表すことがある。
* プロセスの観点から見ると
  - プロセスは、システムに搭載されている主記憶のアドレス空間よりも広いアドレス空間を扱うことができる。
  - 仮想アドレス空間は、プロセスに対して透過的であり、プロセスから見ると連続したメモリ領域である。
  - プロセスは、仮想アドレス空間を介して、他のプロセスが使用中のメモリ領域にアクセスすることはできない。
* システムの観点から見ると
  - システムは、仮想アドレス空間へのアクセスを、論理的に対応する物理メモリアドレスに変換する。
  - 変換先のアドレス空間は必ずしも連続である必要はない。
  - システムは、プロセスが使用中のメモリ領域を、他のプロセスがアクセスできないように保護する。
* 仮想アドレス空間は、その使用者によって以下のように分類されることがある。
  - カーネル空間

    カーネル用に予約された領域。
  - プロセス空間

    ユーザプロセスが使用する領域。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-2. アドレス変換 (ページ管理)	
対応する コースウェア	第 6 回 メモリ管理 (1) 第 7 回 メモリ管理 (2)	

II-6-2. アドレス変換 (ページ管理)

物理メモリに対するアドレス空間と仮想メモリに対するアドレス空間のマッピングとして用いられている「ページ」の概念を解説する。ページングやページテーブルといったページ管理に用いられる基本的な技術を紹介し、ページ管理の効果を説明する。

【学習の要点】

* アドレス空間は、ページと呼ばれる決まった単位に分割されて管理される。
* 仮想アドレス空間のページは、プロセスごとに管理されるページテーブルを基に、アドレス管理ユニットによって物理アドレス空間のページに変換される。
* 変換された物理アドレスに対応するページが存在しない場合は、ページフォルトとしてハードウェアからカーネルに知らされる。ページフォルトを受け取ったカーネルは、適宜必要なページを他のメモリシステムから読み込むなどして配置要求に応える。この操作のことをページングという。

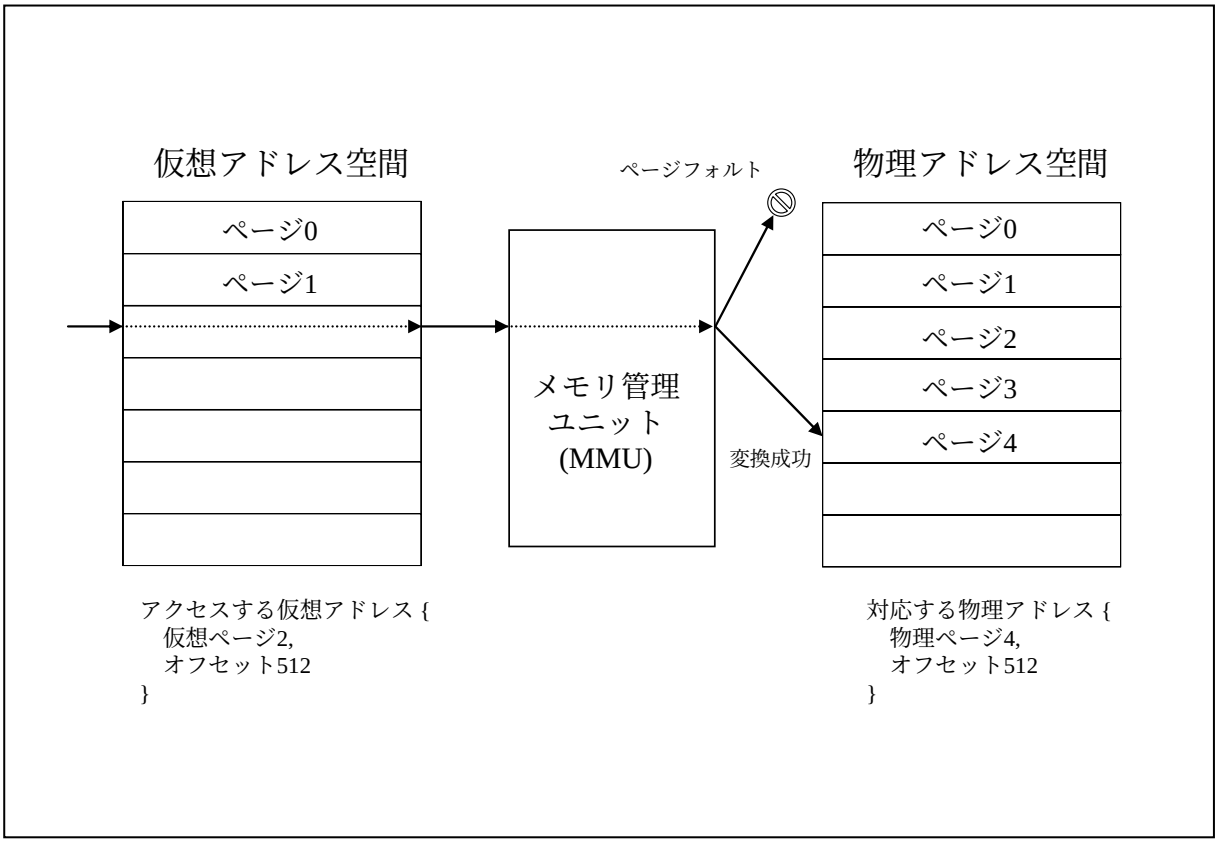

図 II-6-2.　アドレス変換

【解説】

1) ページとは

* アドレス空間は、一定の大きさを持った「ページ」と呼ばれる単位で分割される。ページの大きさ（「ページサイズ」）は、プロセッサのアーキテクチャによって異なるが、必ず 2^n バイトである。
* プロセスは、ページサイズの整数倍の大きさでのみメモリを要求することができる。
* アドレスは、「ページ番号」とそのページ内の「オフセット」で表される。

2) アドレス変換の仕組み

仮想アドレス空間上のページ番号から、物理アドレス空間上のページ番号への対応付け（「アドレス変換」）は、カーネルとメモリ管理ユニット（MMU）が連携して行う。

* ページテーブル
  カーネルは、「ページテーブル」という構造体を管理しており、仮想ページと物理ページの対応を行う。
* TLB (Translation Lookaside Buffer)
  MMU の中には、アドレス変換をより高速に行うために「TLB (Translation Lookaside Buffer)」を持つものがある。TLB は、直近のアドレス変換を保持するキャッシュとして働く。
* アドレス変換とページフォルト
  CPU が仮想アドレスにアクセスすると、まず TLB からページの対応が引かれる。TLB 内に対応が見つからない場合には、「TLB ミス」となり、続いてページテーブルからページの対応が引かれる。対応する物理ページが見つからない場合は、カーネルに「ページフォルト」割り込みが送られる。
* resident 状態
  アドレス変換の結果、要求されたページが物理アドレス空間上に存在する（対応するページが見つかった）場合、そのページは「resident」であるという。ページが resident でない状態とは、不正なメモリアクセスである場合を除いて、物理メモリに空きがないためにそのページが他のメモリシステムに移動されている場合である。

3) ページング

* 前述したように、ページが resident でない場合、カーネルはページフォルト割り込みにより呼び出される。このとき、対応するページは二次記憶など、他のメモリシステムに存在する。メモリアクセスを継続するために、カーネルは他のメモリシステムから物理メモリ上にページを読み込む必要がある。これを「ページング」または「スワッピング」という（もともとスワッピングとは、プロセス全体のページングのこと）。
* ページ置換
  物理メモリに空きがない場合、ページングシステムは「ページ置換アルゴリズム」によって、どのページを「ページアウト」しどのページを「ページイン」するかを決定する。それぞれ、物理メモリから他のメモリシステムへのページの移動、他のメモリシステムから物理メモリへのページの移動の意。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-3. メモリの割り当て方法	
対応するコースウェア	第 7 回 メモリ管理 (2)	

II-6-3. メモリの割り当て方法

カーネル内でのメモリ確保の方法と、そのインタフェースについて説明する。また、カーネル内でのメモリ確保特有の制限や、システムとして提供するメモリ確保のアルゴリズムについても述べる。

【学習の要点】

* カーネル空間は通常ページング不可能である。
* カーネルの使用できるメモリ領域には制限がある。このため、短期的なメモリであっても、現実的にはカーネルが必要とする際には動的に確保する必要がある。
* カーネルがメモリを動的に割り当てるために、必要とされるサイズや高速性要件に適したいくつかの異なるアルゴリズムとそれに対応するインタフェースが用意されている。

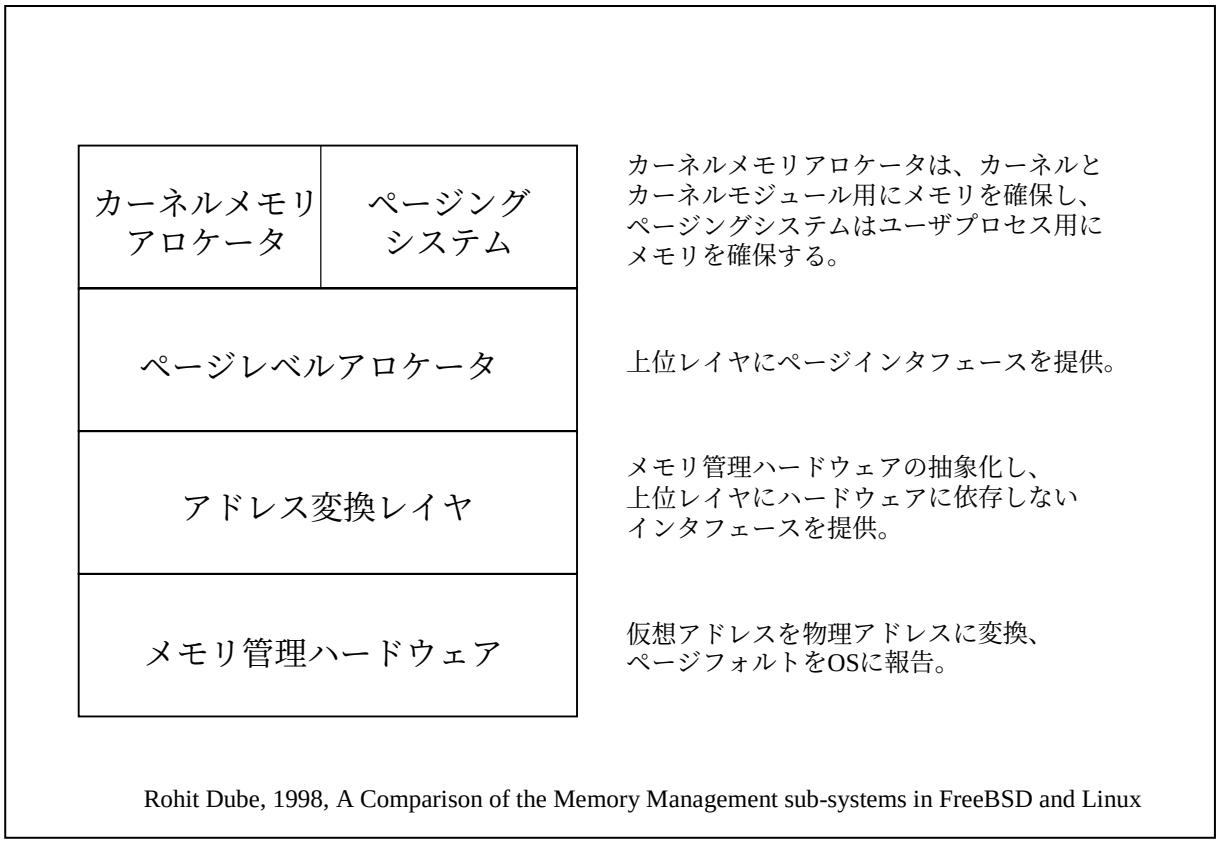

図 II-6-3.　階層化されたメモリ管理システム

【解説】

1) カーネル内での動的メモリ領域確保

* カーネル内では、メモリの使用に関して幾つかの制限がある。
  - 使用するメモリのほとんどはページング不可能である。
  - 通常、アドレス固定の連続した物理メモリ領域を必要とする。
  - 高速性が最も重要となる。多くの場合、カーネルは即座に（処理をブロックすることなしに）メモリ領域を確保する必要がある。

2) カーネル内での動的メモリ領域確保用インタフェース

* ユーザプロセスは、malloc(), free()システムコールを用いて動的なメモリ領域の確保を行う。カーネル内でも、動的にメモリ領域を確保するために、malloc()と似たインタフェースが用意されている。Linux では以下のインタフェースが使用できる。
  - kmalloc(), kfree()
    カーネルが連続した物理メモリ領域を確保するために使用。物理メモリ上に連続した領域を確保することで、空間的局所性が得られ、TLB を最大限活用できるため高速である。
  - vmalloc(), vfree()
    仮想メモリ領域から仮想的に連続したメモリ領域を確保するために使用。kmalloc(), kfree()と異なり、物理メモリ領域上では連続しているとは限らない。このため、速度は劣るものの大きな領域を確保することができる。

3) メモリ確保の構成要素

* ページレベルアロケータ
  ページレベルアロケータは、ハードウェアによって行われたアドレス変換の結果得られたページをカーネルメモリアロケータとページングシステムに渡すインタフェースを提供する。

4) ページングシステム

ページングシステムは、仮想メモリシステムで用いられる、「デマンドページング」や「スワッピング」ポリシーを実装するものである。主にユーザプロセスに対してメモリ確保の仕組みを提供する。

* カーネルメモリアロケータ
  - 2 の累乗フリーリスト
    ページ単位よりも小規模なメモリを確保したい場合、ページングシステムでは内部断片化が起こる。このような場合、ページを 2^n の大きさを持った小さなゾーンに分割する戦略が取られる。これは「2 の累乗フリーリスト」として知られる。これを発展させたものに「バディシステム」がある。
  - スラブアロケータ
    Linux には、さらに効率的（断片化を極力抑える）にメモリ領域を確保できる「スラブアロケータ」が用意されている。これは、オブジェクトに対してスラブと呼ばれる連続した物理メモリ領域を割り当て、この領域を次回同種のオブジェクトに対するメモリ要求があった場合に再利用するものである。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-4. プロセス空間の管理方法	
対応するコースウェア	第 8 回 メモリ管理 (3)	

II-6-4. プロセス空間の管理方法

プロセス空間を管理する方法の具体的な技術について説明する。スワップの概念を示し、ページキャッシュ、ページフォルト、デマンドページングといった個々の概念を解説する。また関連する話題として実メモリが不足してきた際に処理が必要となるガベージコレクション等の技術についても述べる。

【学習の要点】

* プロセスには仮想アドレス空間が割り当てられ、プロセス空間と呼ばれることがある。それぞれのプロセスに割り当てられたアドレス空間は独立しており、メモリ管理ユニット (MMU) によって互いに保護される。
* プロセスが物理アドレス空間の空きページを超えるような仮想ページを要求した場合、メモリ管理ハードウェアはページフォルトを検出し、カーネルに知らせる。このときカーネルはページングにより必要なページを確保する。これをデマンドページングという。
* 一旦、二次記憶からメモリページにデータが読み込まれると、二度目のアクセス時にそのページがそのまま利用可能であればそのままプロセスにそのページを返す。この仕組みをページキャッシュという。

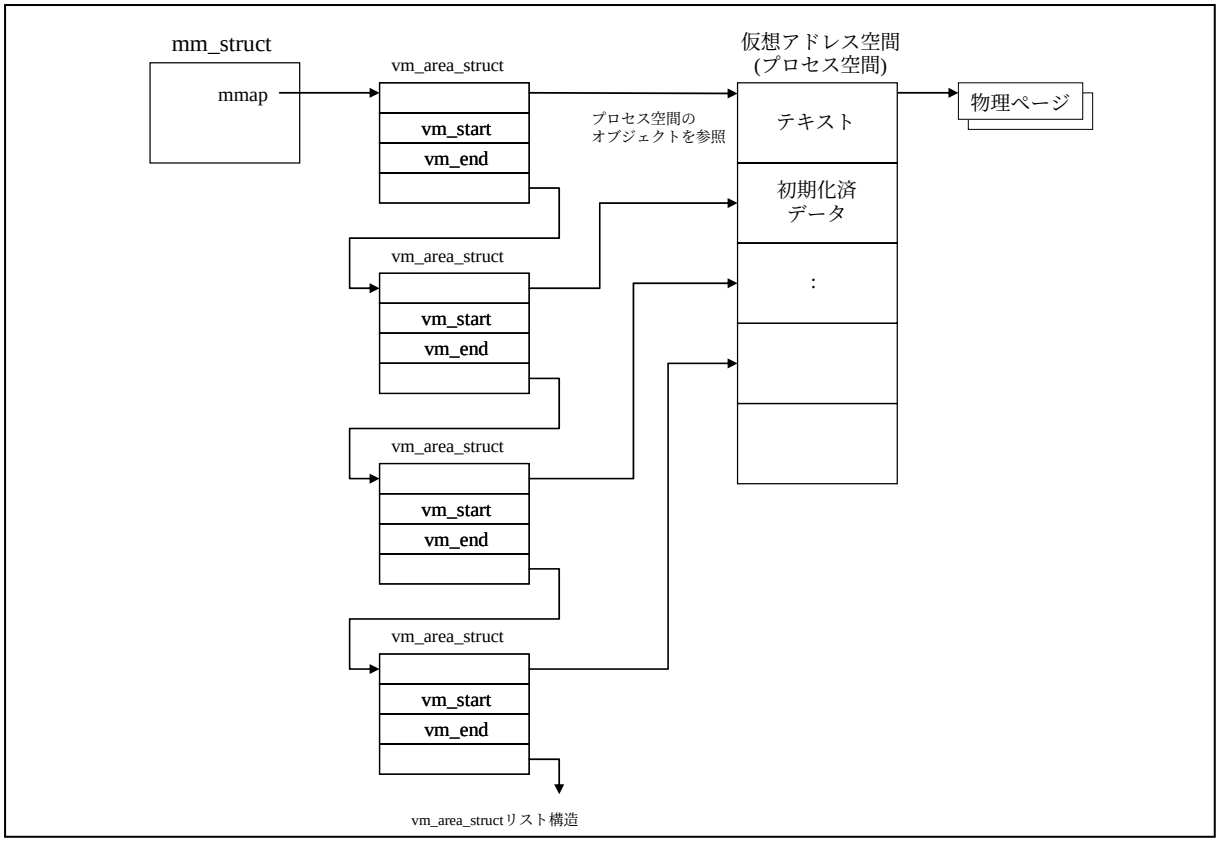

図 II-6-4.　プロセス構造体

【解説】

1) プロセス空間

* プロセスには独自の仮想アドレス空間が割り当てられる。これを「ユーザ仮想アドレス空間」または単に「ユーザ空間」、あるいは「プロセス空間」と呼ぶ。
* Linux において、プロセス空間は mm_struct 構造体で表現される。mm_struct 構造体の mmap メンバは、vm_area_struct 構造体のリストで、プロセスの実行に必要なプロセス空間上のオブジェクトを表現する。オブジェクトは一つ以上の物理メモリページに対応する。
* 物理ページへの変換は MMU により行われる。この時、他のプロセスが使用中のアドレス空間へのアクセスも禁止される。
* vm_area_struct 構造体は、プロセス空間中の開始アドレスと終了アドレスを保持する。
* 必要な vm_area_struct 構造体に素早くアクセスするために、mm_struct 構造体は、vm_area_struct 構造体のリストだけではなくレッドブラックツリー構造も保持する。

2) メモリの効率的な利用

* デマンドページング
  プロセスが実行するためには物理メモリ領域が必要であるが、起動時点でプロセス全体が resident (II-6-2 参照) である必要はない。その他のページはページフォルトが発生した際に必要に応じて「スワップ領域 (物理ページが存在する物理メモリ以外の場所)」から読み込まれる。これを「デマンド (要求時) ページング」と呼ぶ。
  - 利点
    起動時にすべての物理ページを読み込む必要がないのでメモリ領域を節約でき、起動も速い。
  - 欠点
    初めてアクセスされるページは、ページフォルト処理や読み込みのコストがかかるため遅延が発生する。
* ページキャッシュ
  一旦スワップ領域から物理メモリ領域に読み込まれたページのうち、読み込み専用で共有可能なものは、ページアウトせずに再利用することができる。これによってページイン、ページアウトにかかるコストを押さえることができる。この仕組みを「ページキャッシュ」という。
* スワップデーモン
  ページキャッシュを用いると、プロセスが必要とするページにアクセスする速度を向上させることができるが、キャッシュが溜まるにつれて物理メモリ領域を圧迫することになる。このように、物理メモリ領域上に空きページが少なくなるのを監視するのが「スワップデーモン」であり、Linuxでは「kswapd」が相当する。
  空きページの不足を検知すると、スワップデーモンはページキャッシュを縮小したり、ページアウトしたりすることで空きページを確保する。直接カーネルシステムと関連はないが、オブジェクト指向言語で実装されることの多い「ガベージコレクタ」の動きはスワップデーモンの動きに似ている。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-5. 仮想ファイルシステム	
対応するコースウェア	第 9 回 ファイル管理 (1) : 仮想ファイルシステム	

II-6-5. 仮想ファイルシステム

ファイルシステムの違いを意識することなくあらゆるファイルシステムへ一元的にアクセスできるようにするための仮想ファイルシステムについて解説する。仮想ファイルシステムの概念と特徴を説明し、実際に操作がどのように行われるかについて示す。

【学習の要点】

* 仮想ファイルシステムは、ファイルシステムの上位層に存在し、異なるファイルシステムに対して透過的な操作を実現する共通インタフェースである。
* よく知られているファイルシステムの実装には、proc、Linux 標準のローカルファイルシステムである ext3、ネットワークファイルシステムでは NFS や CIFS などがある。Linux においてこれらのファイルシステムへの操作は、システムコールから呼び出され、仮想ファイルシステムインタフェースを介してアクセスされる。
* 仮想ファイルシステムインタフェースを持つファイルシステムは新しく定義することができる。仮想ファイルシステムのノードは、実ファイルシステムへのポインタを持つように設計する。このポインタは、ローカルファイルシステムの場合は inode を参照する。

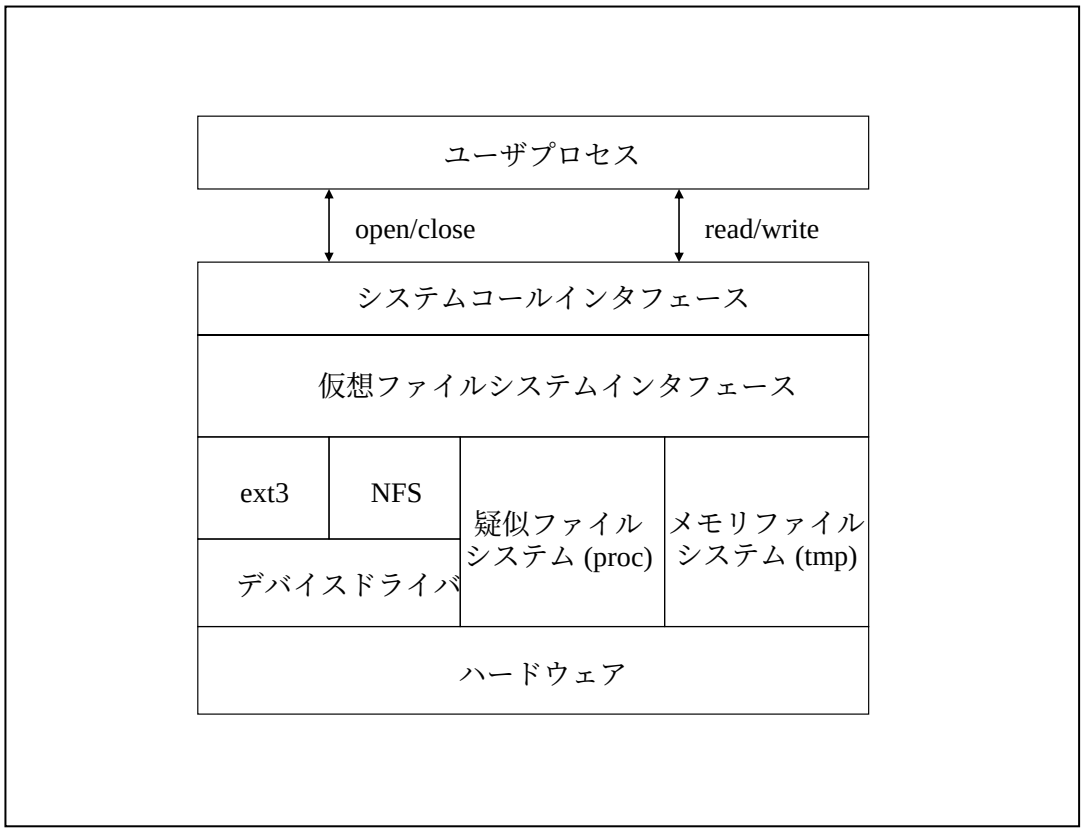

図 II-6-5.　仮想ファイルシステムインタフェース

【解説】

1) 仮想ファイルシステムインタフェース

* Linux では、異なるファイルシステム実装に対して画一的なシステムコールでアクセスできるよう、仮想ファイルシステム (VFS) 層が設けられている。
* VFS 層を導入することで、ローカルファイルシステムやネットワークファイルシステム (NFS) といった違いを意識せずに、共通のインタフェースを使ってファイルの読み書きを行うことができる。
* VFS のおかげで、ファイルシステムへのアクセスと同様のシステムコールを用いてカーネル内部の情報をユーザに提供することができる。このような仕組みは疑似ファイルシステムと呼ばれ、Linux の procfs はこの例のひとつである。procfs は通常/proc にマウントされる。
* tmpfs は、仮想記憶常駐型のファイルシステムである。ファイルはカーネル内のキャッシュ (ページキャッシュとスワップ) に置かれ、揮発性である。tmpfs は通常/tmp にマウントされる。

2) 仮想ファイルシステムの構成要素

* 仮想ファイルシステムは、ファイルシステムの実装に依存しない抽象的なインタフェースを提供する。それぞれのインタフェースの関係を表す代表的なメンバを紹介する。
  - namespace 構造体
    プロセスの属する名前空間を表す構造体。名前空間は、マウント状態 (vfsmount 構造体で表されるファイル階層データ) の集合を表し、vfsmount 構造体リストのポインタを保持する。
  - vfsmount 構造体
    マウント状態を表す汎用的な構造体。自身の属する名前空間のポインタと、ファイルシステムスーパーブロック (super_block 構造体で表される) のポインタを保持する。
  - super_block 構造体
    ひとつのファイルシステムに関する情報を表す汎用的な構造体。ファイルシステム固有情報のポインタを保持する。
  - inode 構造体
    ファイルシステム内のひとつのファイルに関する情報を表す汎用的な構造体。自身の属するファイルシステムスーパーブロックのポインタ、ファイルシステム固有情報のポインタを保持する。
  - dentry 構造体
    ファイルの親子関係とその名前を表す構造体。i ノード (inode 構造体) のポインタを保持する。
  - file 構造体
    プロセス毎に、ファイルをオープンしたときに生成される構造体。ディレクトリエントリ (dentry 構造体) のポインタを保持する。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-6. カーネルによるファイル操作	
対応する コースウェア	第 10 回 ファイル管理 (2) ：ファイルの操作	

II-6-6. カーネルによるファイル操作

ファイルシステムに対する一般的な操作を説明する。ファイルのオープンとクローズ、データの読み書きなど基本的な操作から、ディスクへの書き出しがどのように行われるか、同期がとられるタイミングといったカーネルの内部動作についても解説する。

【学習の要点】

* ユーザプロセスは、open や socket といったシステムコールを介してファイル記述子を獲得し、カーネル内のオブジェクトにアクセスする。
* カーネルはユーザプロセスから指定されたファイル記述子からファイル構造体を取得する。ファイル構造体は open をはじめとするシステムコール毎に割り当てられ、以降の対応するシステムコールの呼び出しによって、読み込み、書き込み、ファイル状態取得、ファイルのクローズなどをサポートする。
* ファイル構造体は、ファイルのオフセットを保持する。そのため、異なる open システムコールによって割り当てられたファイルオフセットはそれぞれ独立して管理される。read, write システムコールは、ファイル構造体の管理するオフセットからそれぞれ処理を開始する。

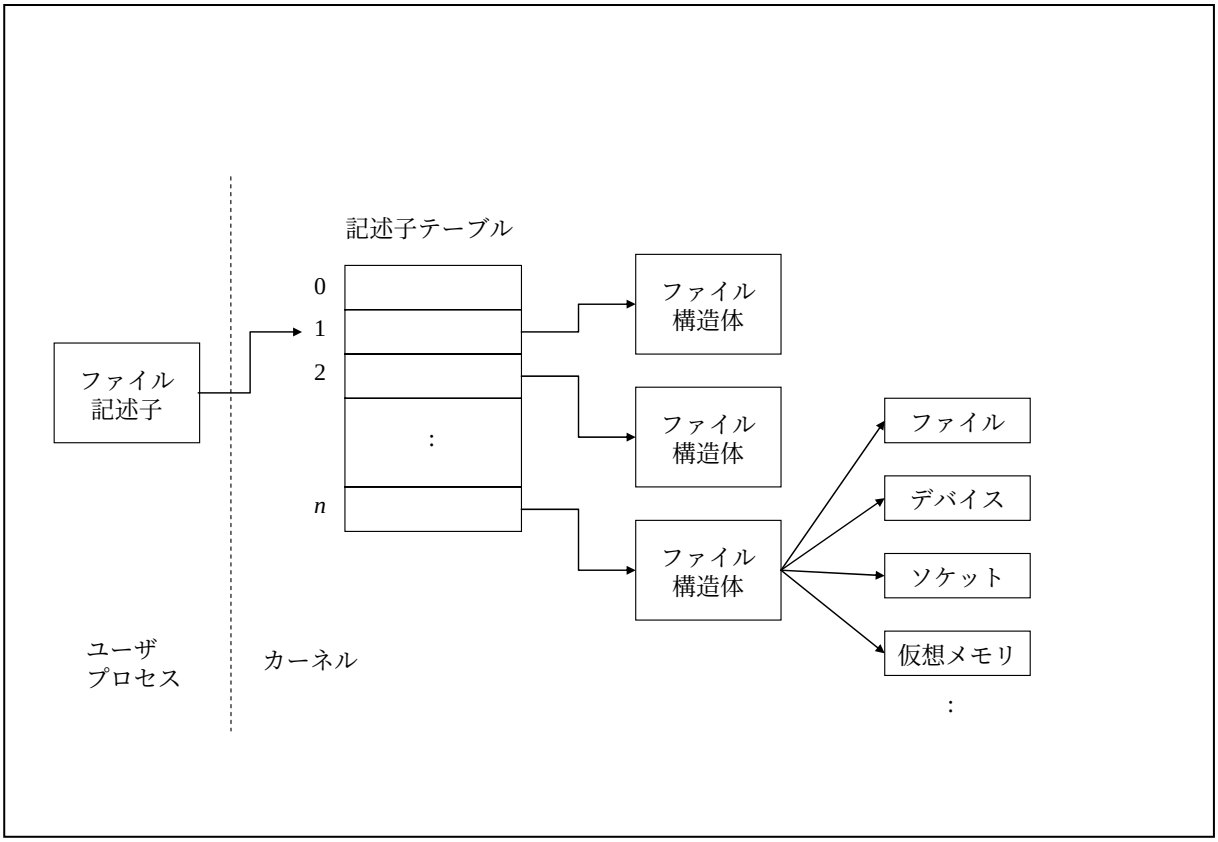

図 II-6-6.　記述子管理と参照関係

【解説】

1) ファイル記述子

* ユーザプロセスが入出力操作を行う際は、ファイル記述子を通して行われる。
* ユーザプロセスは、open, socket など、オブジェクトの種類に合わせて用意されたシステムコールによって、ファイル記述子を取得することができる。
* カーネルは、プロセス毎に記述子テーブルを管理する。記述子の値は、このテーブルのインデックスとなっており、テーブルの値はファイル構造体のインスタンスを指す。

2) ファイル構造体

* ユーザプロセスが一旦ファイル記述子を取得すると、その後の操作は、取得した記述子を指定したシステムコールによって行われる。カーネル内では、記述子に対する操作はファイル構造体に対する操作として扱われる。
* ファイル構造体のインスタンスは、記述子の対応するカーネル内の実体オブジェクトを指す。実体オブジェクトは、ファイルであったりソケットであったり、メモリマップされたファイルであったりする。
* ファイル構造体は記述子と同様にプロセス毎に生成される。ファイル構造体は、対応するオブジェクトのオフセット（そのプロセスが、記述子に対して次の read や write で読み書きを開始する位置）を保持する。オフセットの値は、read や write を行う度に更新される。
* ファイル構造体のオフセットを直接指定するには、lseek システムコールを用いるが、すべての実体オブジェクトがこれをサポートしているわけではない。ソケットやパイプ、FIFO に対する lseek システムコールは常に失敗する。lseek システムコールは、ランダムアクセス可能なオブジェクトでのみ利用可能である。

3) バッファと同期

* ユーザプロセスが自身のプロセス空間にファイルの内容を読み込んだり、書き出したりする際には、通常カーネルバッファを経由して行われる。
* カーネルバッファを経由せずに入出力を行うこともできる。直接転送は、ユーザプロセスの書き込みが、カーネルバッファを経由せずに直接ディスクに対して行われるようにする。また、mmap によってプロセス空間にファイルをマップすることで、ユーザプロセスは自身のプロセス空間への読み書きを通してファイルにアクセスできる。
* open や fcntl システムコールでは、ファイル記述子を通して間接的にファイル構造体の振る舞いを変更するためのフラグを設定できる。そのうち、以下にあげるコマンドはファイルの読み書きのポリシーを決定する。
  - O_SYNC（同期書き込み），O_DIRECT（直接転送），O_ASYNC（非同期書き込み），O_NONBLOCK (ノンブロッキング I/O)

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-7. 特別なファイル	
対応するコースウェア	第 11 回 ファイル管理 (3) :特殊ファイル	

II-6-7. 特別なファイル

ブロックデバイスやパイプ、proc ファイルシステムのような疑似ファイルシステムなど、Linux カーネルからファイルシステムインタフェースによりアクセスすることができる特別なファイルを説明し、その意義とメリットについて触れる。

【学習の要点】

* 外部のデバイスとの入出力を、ファイルシステム上のファイルと同じシステムコールを用いて行えるように、カーネルは対応するデバイス用のインタフェースとして、特殊なファイルを用意する。これをスペシャルファイルという。
* スペシャルファイルへの入出力は、カーネル内部でデバイスドライバに転送される。
* デバイスドライバは、2 つのインタフェースを提供する。キャラクタデバイスインタフェースは、通常のファイルと同様に、バイト列をそのままデバイスに転送する。ブロックデバイスインタフェースは、バイト列をデバイスの要求する構造に変換してから転送するため、デバイスの種類に合わせて適切なバッファリングが必要。
* スペシャルファイルは必ずしもデバイスと関連づいている必要はない。FIFO は名前付きパイプを実現するために使用されるスペシャルファイルである。

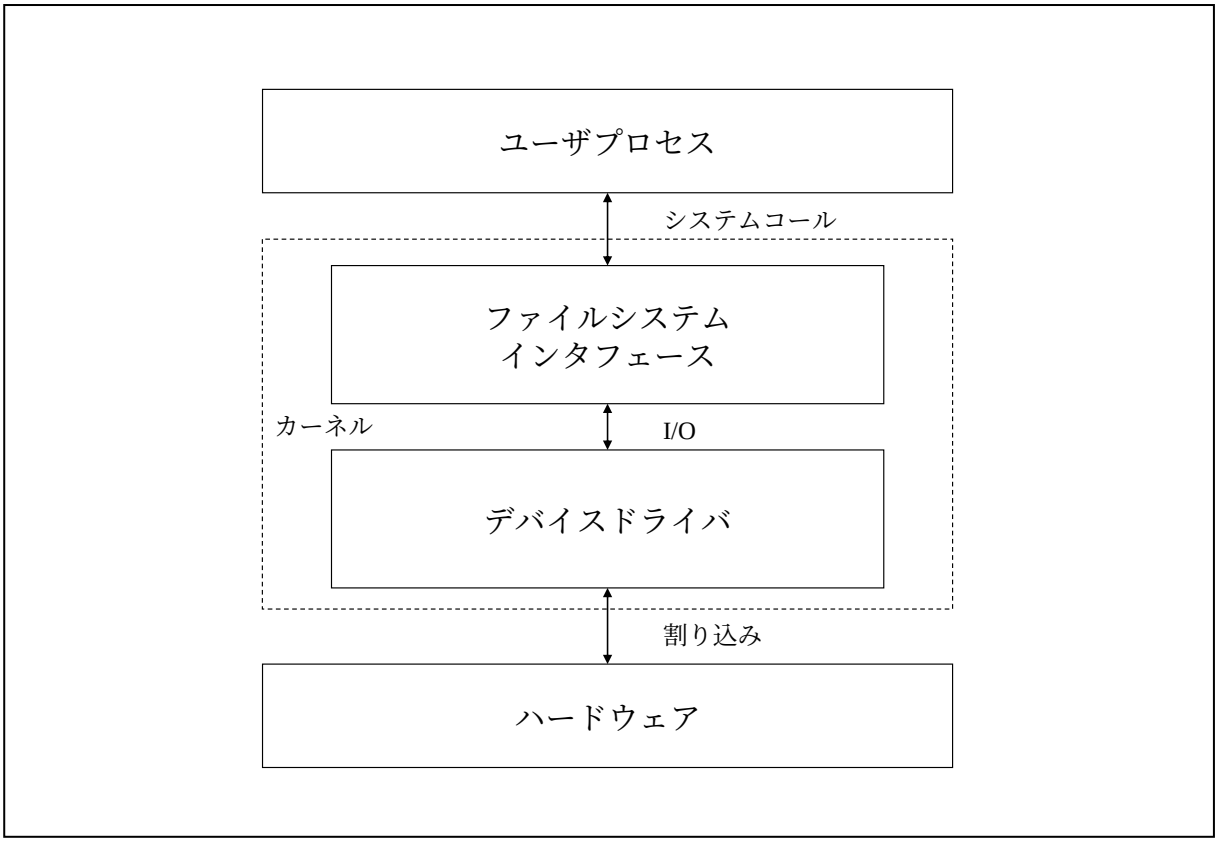

図 II-6-7.　デバイスドライバ

【解説】

1) デバイスドライバ

* カーネル内部では、ハードウェアデバイスの特性はデバイスドライバで表される。デバイスドライバは、ハードウェアデバイスからの割り込み処理と、ユーザプロセスまたはカーネルからの入出力要求を処理する。
* ハードウェアデバイスは、その入出力の種類によってブロックデバイスとキャラクタデバイスに分けられる。
  - ブロックデバイス
    ランダムアクセスの可能なデバイス。構造的な入出力を要求する。
  - キャラクタデバイス
    シーケンシャルアクセスのみが可能なデバイス。入出力は、非構造的なバイト列で行われる。
* ハードウェアデバイスは、カーネルによってファイルシステム上のファイルとして表現される。このようなファイルをスペシャルファイルという。
* ユーザプロセスは、スペシャルファイルに対するシステムコールによってハードウェアデバイスを操作することになる。
* デバイスドライバは、ユーザプロセスの呼び出す read, write, poll といったシステムコールに対応するエントリポイントを実装する。
* デバイスドライバは、必ずしもハードウェアデバイスと通信する必要はない。/dev/null (出力を破棄) や、/dev/random (乱数を発生) は、疑似デバイスと呼ばれる。

2) 疑似ファイルシステム

* 疑似ファイルシステムとは、ユーザプロセスがそれにアクセスしたときに、特殊な結果を返すよう設計されたファイルシステムである。
* Linux では、スペシャルファイルやソケットの機能を実現するのに、内部的に疑似ファイルシステムを使用する。
* ユーザプロセスから見て最も身近なファイルシステムに、proc ファイルシステムと sys ファイルシステムがある。どちらのファイルシステムも、カーネルの内部で保持する情報を、ユーザプロセスがファイルシステムインタフェースを通してアクセスすることを可能にする。
* proc ファイルシステムは、通常/proc にマウントされる。proc ファイルシステムは実行中のプロセスの状態を表示するのに便利である。これらは、/proc 以下のプロセス番号の名前を持つディレクトリとして表現される。
* /proc/meminfo や/proc/cpuinfo は、それぞれ仮想記憶システム状態、CPU 情報を表示させるためにしばしば使用される。
* sys ファイルシステムは、Linux カーネル 2.6 から導入された。従来 proc ファイスシステムを通して得られた情報のうち、プロセスに関わらない部分はこちらに移された。sys ファイルシステムも、一般的なファイルと同様のインタフェースでアクセスすることができる。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-8. プロトコルスタック	
対応するコースウェア	第 12 回 ネットワーク(1)：ソケットインタフェース	

II-6-8. プロトコルスタック

OSI の参照モデルについて説明し、TCP/IP との対応関係を示す。また、Linux のソケットインタフェースからどのようにネットワークが取り扱われるか、Linux カーネルとネットワークのインタフェースについての説明を加える。

【学習の要点】

* ISO は、開放型システム相互接続(OSI)参照モデルを提唱している。Linux の TCP/IP 実装は OSI に完全準拠したものではない。
* カーネル内部のネットワークシステムは、トランスポート層、ネットワーク層、リンク層に対応を取って考えることができる。
* ネットワークシステムへは、ソケット API を通してユーザプロセスからアクセスされる。ソケット API は、ユーザプロセスに、ネットワークアーキテクチャに依存しない透過的なインタフェースを提供する。
* Linux カーネルは、ネットワークデバイスドライバを通してハードウェアにアクセスし、ネットワーク層にデータグラムを渡す。

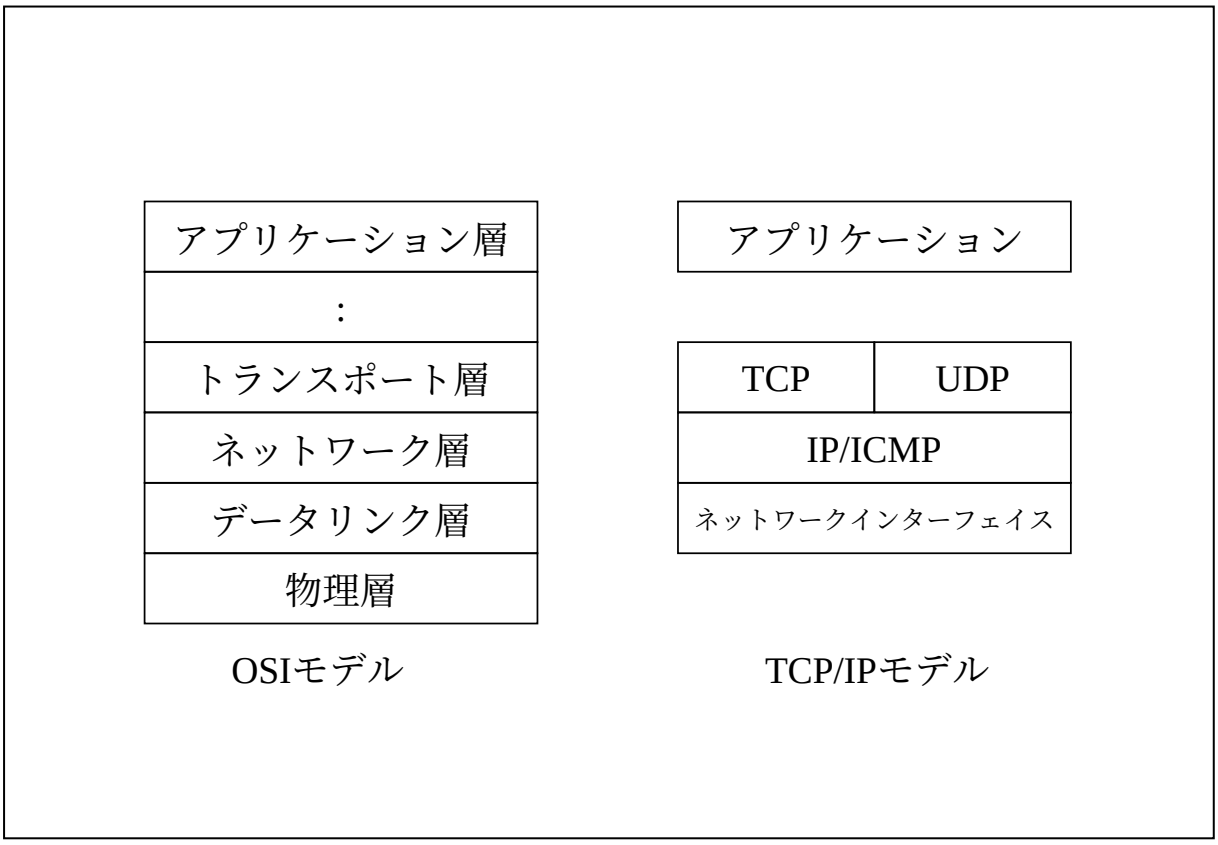

図 II-6-8.　OSI モデルと TCP/IP モデルの対応

【解説】

1) OSI モデル

* ISO は、ネットワーク機能を7階層に分けて分類した開放型システム相互接続（OSI）参照モデルを提唱している。それぞれの階層は、アプリケーション層、プレゼンテーション層、セッション層、トランスポート層、ネットワーク層、データリンク層、物理層である。
* Linux カーネル内のネットワークシステムは、概ね OSI モデルのトランスポート層、ネットワーク層、データリンク層にそれぞれ対応を取ることができるが、標準化されたものではない。

2) TCP/IP モデル

* 「TCP/IP」は、IETF によって標準化されている「インターネット・プロトコル・スイート」のうち、最も重要な 2 つのプロトコルの組を表したもので、関連するプロトコル群を代表する呼称である。
* Linux カーネルのネットワークシステムとそのアプリケーションプロトコルは、「インターネット・プロトコル・スタック」の、アプリケーション、トランスポート、インターネット、ネットワーク・アクセスの各階層を実装したものと言える。

3) ソケット API

* ソケット API は、ユーザプロセスがカーネル内のネットワークシステムにアクセスするためのインタフェースである。Linux でソケットを生成するには、ライブラリ関数を通して間接的にシステムコールにアクセスする。
* カーネル内部でソケットが確保されると、ユーザプロセスはソケットを表すファイル記述子を獲得する。以降ソケットに対する操作は、獲得した記述子を通して行うことになる。

4) ネットワークインタフェース

* カーネルを中心に考えると、ソケット API はユーザプロセス側のインタフェースである。これに対して、ネットワークインタフェースは、ハードウェア（ネットワークデバイス）側のインタフェースと言える。
* ユーザプロセスからソケット API を通して送り出されたデータ（外向き）は、ヘッダを付加されながら順々にプロトコルスタックを降りていき、最終的にネットワークインタフェースを通してネットワークデバイスに渡される。
* ネットワークからデータを受信すると（内向き）、カーネルはネットワークデバイスからの割り込みを処理する。受信したデータは、ネットワークインタフェースを経由してキューに入れられる。各階層においてパケットヘッダの解析が行われ、最終的にはユーザプロセスの保持するファイル記述子から読み込み可能となる。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-9. IP ネットワークの基本	
対応するコースウェア	第 13 回 ネットワーク (2) : IP と UDP	

II-6-9. IP ネットワークの基本

IP ネットワークの基本的な概念と、IP ネットワークで利用される基本的な要素技術について解説する。IP パケットの構成やルーティングの概念、送受信がどのように行われるかといった基本的な技術について説明する。

【学習の要点】

* インターネットプロトコル (IP) には、IPv4 と IPv6 の二つのバージョンが定義されている。今日一般的に IP というと、IPv4 を指す。
* IP は OSI モデルで言うとネットワーク層、インターネット・プロトコル・スタックではインターネット層に実装され、パケット交換ネットワークである IP ネットワークを構成することを可能にする。
* 普及している IP ネットワークは、ホストがあるネットワークに接続され、ネットワーク同士がルータを介して Ethernet で接続されているモデルが一般的である。
* IP は、ホストのアドレスを管理し、データ送信経路の制御を行う。

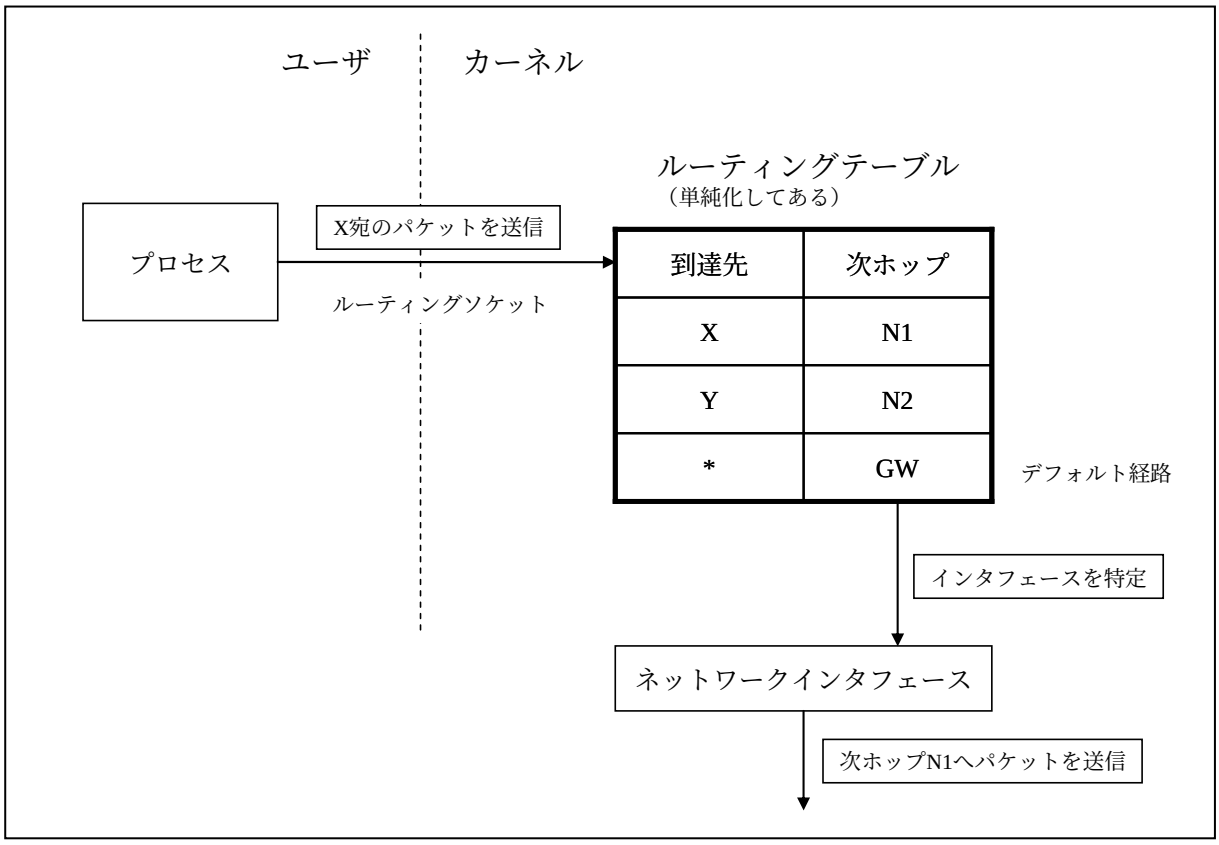

図 II-6-9.　パケットを送信する際のルーティング

【解説】

1) IP

* IP (インターネットプロトコル) は、パケット交換相互コンピュータネットワークにおいて、データグラムを送受信するためのプロトコルである。
* データグラムは、ホストからホストへ送信される。発信元と送信先は、決まったデータグラム構造をやりとりし、お互いを識別する。この決まったデータグラム構造を IP パケットという。IP パケットには、識別子となる IP アドレスが含まれる。IP アドレスによって相互に接続されるネットワークを IP ネットワークということがある。
* IP アドレスには、2 のバージョンが存在する。現在多く使用されているのはバージョン 4 (IPv4 アドレス) である。IPv4 の後継として、バージョン 6 (IPv6) が定義されている。

2) ルーティング

* IP ネットワークは、IP パケットを相互に交換するネットワークであり、それぞれのネットワーク同士は IP パケットの経路を制御 (ルーティング) するルータによって接続される。
* IP ネットワークにおけるルーティングとは、IP パケットに含まれる発信元アドレスや送信先アドレスによって、そのパケットを渡す次のホストを決定することである。
* ルーティングは、ルーティングテーブルを元に行われる。ルーティングテーブルのエントリには、次ホップのアドレス、ネットワークインタフェース情報、経路メトリックなどといった情報が含まれる。Linux の場合、ルーティングテーブルはカーネル内のネットワークシステムによって管理される。
* ユーザプロセスは、カーネル内で管理されるルーティングテーブルを変更するためにソケットインタフェースを利用する。

3) ICMP

* ICMP は、IP の制御情報とエラーメッセージをやり取りするプロトコルである。ICMP メッセージは、主にカーネル内のモジュールによって送受信される。

4) IPv6 アドレス

* IPv4 アドレスは、32 ビットのアドレス空間を構成する。この空間上に存在できる要素数は、2^{32} 個である。これに対して、IPv6 は、128 ビットのアドレス空間を構成する。IPv6 アドレス空間に存在できる要素数は 2^{128} 個となり、IP アドレス枯渇の懸念は一気に解消される。
* IPv6 は、インターネット層の IPv4 アドレス実装を置き換える。また、プロトコル仕様にセキュリティプロトコル IPSec を含むほか、オートコンフィグレーションプロトコルも含む。

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	6　Linux カーネルに関する知識 II	応用
習得ポイント	II-6-10. UDP と TCP	
対応する コースウェア	第 14 回 ネットワーク (3) : UDP と TCP 第 15 回 ネットワーク (4) : TCP フロー制御と輻輳制御	

II-6-10. UDP と TCP

インターネットで広く利用されている UDP による通信方式と、信頼性を高めた TCP による通信方式について説明する。またとくに TCP に関しては通信手順の詳細を示し、さらにフロー制御や輻輳制御など高度な通信制御技術も示す。

【学習の要点】

* IP 上に定義された TCP, UDP プロトコルは IP の接続先情報 (IP アドレス) に加えて、ポート番号を使用する。
* UDP は、送信元、送信先情報とチェックサム機能のみを提供する単純なデータグラムプロトコルである。
* TCP は、UDP にはない高度な機能を提供する。TCP は、明示的な接続、切断と接続の状態を管理する。接続の手続きは、3 ハンドシェイクによって行われ、情報が確実に交換できることを保証する。
* UDP は信頼性を担保しないが、TCP はこれを保証する。また、フロー制御、帯域外通信、輻輳回避機能も含まれる。

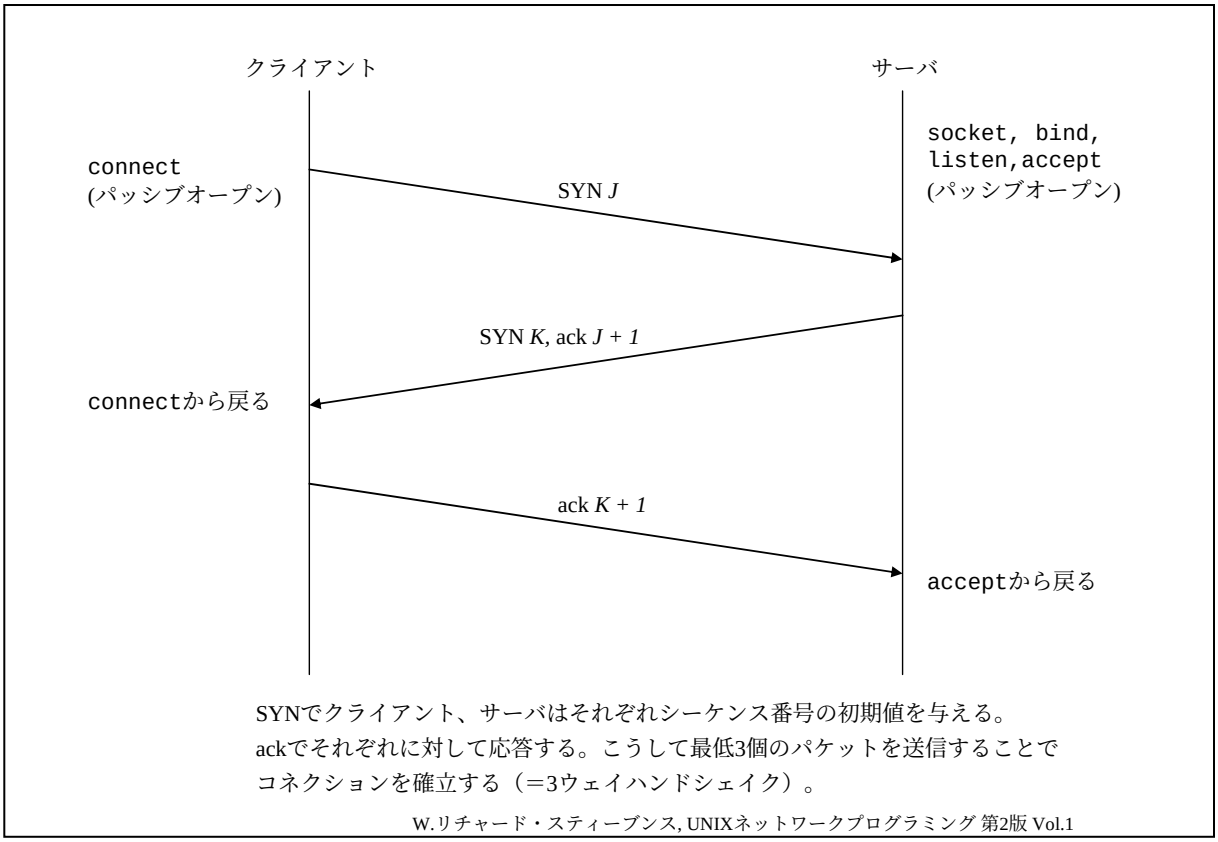

図 II-6-10.　TCP の 3 ウェイハンドシェイク

【解説】

1) IP とポート番号

* IP パケットは、発信元ホストアドレスと送信先ホストアドレスを格納する。ホストに到達した IP パケットを、ホストで動くどのプロセスが処理するかを決定するには、ポート番号を使用する。
* UDP と TCP は、トランスポート層のプロトコルである。ポート番号は、トランスポート層のプロトコルヘッダ内に格納される。IP ヘッダには、トランスポート層で使用するプロトコル番号が格納される。
* Linux ネットワークシステムは、ソケットによってプロセスとそのプロセスに関連づいたポート番号を管理する。

2) UDP

* UDP (User Datagram Protocol) は、トランスポート層で実装される最も単純なメッセージ指向プロトコルである。UDP は、信頼性を保証しないプロトコルであり、配送の確実性と、順序および重複がないことは保証されない。
* UDP プロトコルヘッダは、発信元ポート番号、送信先ポート番号、パケット長、チェックサムのみが含まれる。
* UDP を用いた有名なアプリケーションプロトコルとして、DNS (Domain Name System), SNMP (Simple Network Management Protocol), DHCP (Dynamic Host Configuration Protocol) がある。

3) TCP

* TCP (Transmission Control Protocol) は、トランスポート層で実装される接続指向のプロトコルである。UDP と違い、信頼性を保証するがプロトコル仕様は複雑であり、遅延が致命的であるアプリケーションには向かない。
* TCP で最も重要な機能
  - 明示的な接続の開始と終了
    ユーザプロセスは、connect を使用して明示的に接続を開始する。接続手続きは、3 ウェイハンドシェイクと呼ばれ、双方で SYN と ACK を適切な順序で送り合うことで、確実に接続の成功／失敗を上位層に知らせる。
  - 信頼性と順序および重複がないことの保証
    TCP 接続はネットワークを介したストリームをオープンする。ストリームは、順序を持つデータ列の修辞である。TCP ヘッダにはストリームを実現するためのシーケンス番号が含まれ、これによって信頼性と順序性を保証する。
* その他の TCP の機能
  TCP は通信の信頼性を保証し、以下に示す高度な通信制御技術も提供する。
  - フロー制御…受信側のバッファサイズに合わせて送信データを調整する
  - 帯域外通信…できるだけ早く処理すべき緊急データの送信をサポート
  - 輻輳回避…未知のネットワークに対して不適切に大量のデータを送信しないスロースタートなど、いくつかのアルゴリズムをサポート