25. 組み込み開発環境に関する知識 II

1. 科目の概要

組み込みシステムの開発手法と組み込みシステムを開発するための開発環境について、典型的な開発環境や実際の利用例を解説する。組み込みアプリケーション開発のポイントを示し、クロス開発環境、GNU 開発環境、組み込み Linux といった具体的な開発環境を紹介する。

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-25-1. 組み込みアプリケーション開発事例	組み込みアプリケーション開発の事例として、キャッシュとROM化の取り扱い、マイコン周辺機能、周辺I/O制御プログラミング、DRAM制御プログラミングといった具体的な事例を示し、またそれぞれについての仕様や開発環境、開発の実際について解説する。	9
II-25-2. 組み込みアプリケーション開発における留意点	スタートアップルーチンの実装や例外処理、安全化設計、ROM化において気をつけるべき点、低消費電力モード、Watch Dog Timer (WDT)の利用など、組み込みアプリケーション開発において留意すべき様々な事項を紹介する。	9
II-25-3. クロス開発環境の種類と特徴	組み込みアプリケーションを開発するためのクロス開発環境についてその全体像と構成、特徴、必要性等を説明する。さらにクロス開発ツールの種類として半導体ベンダが提供するもの、サードパーティが提供するもの、Free Software Foundationの提供によるGNU開発環境などがあることを挙げ、それぞれを簡単に紹介する。	10
II-25-4. クロス開発環境を利用した組み込みシステム開発	クロス開発環境を利用した組み込みアプリケーションの開発について、開発環境の選択方法や開発環境の構築、クロス開発環境を活用した実際のアプリケーション開発手順等を解説する。さらに複数のターゲットを対象としたクロス開発やライセンスの取り扱いなど、関連する事項についても触れる。	10
II-25-5. GNU開発環境の利用方法	GNU開発環境の全体像と構成、特徴、必要性について説明する。またGNU開発環境を用いた開発の手順や、それ以外の開発環境との比較、GNU開発環境を利用した際におけるライセンスの取り扱いやサポート等、関連するトピックについても触れる。	11
II-25-6. GNU開発環境を構成するツール群	GCCやGDBといった基本的なツールに加え、as (アセンブラ)、ld (リンカ)、nm (シンボルテーブルの表示)、objdumpやobjcopyといったオブジェクトファイルを操作するツール、オブジェクトファイルからシンボルを削除するstripなど、binutils (バイナリユーティリティ)に含まれる各種のツールを説明する。	11
II-25-7. GNU開発環境を用いたアプリケーション開発	GNU開発環境による組み込みアプリケーション開発の手順について、その特徴と役割を概説する。またGNU開発環境の入手方法、環境構築の方法、GDBスタブのROM書き込み、アプリケーションのコーディングとビルドおよびデバッグといった具体的な手順の内容を説明する。	12
II-25-8. 組み込みLinuxの開発環境	組み込みLinuxを用いた開発環境について説明する。その全体像と構成、組み込みLinux開発環境を利用する必要性を解説する。またARM、PXA、SHなど、CPUの系列別に組み込みLinuxベンダの対応状況を紹介する。	13
II-25-9. 組み込み開発環境の評価	購入や保守にかかる費用、操作性の善し悪し、操作方法の習得にかかる期間、前提とする知識、ターゲットシステムのハードウェアとの適合性、他ツールとの接続性や互換性等、組み込みアプリケーション開発環境を評価する項目と評価方法について説明する。	14
II-25-10. 組み込み開発環境におけるデバッグの実際	組み込み開発環境において、プログラムをデバッグする手順を解説する。割り込み処理のデバッグ方法や、メモリ保護エラー、表示装置以上、タイマーの不具合、通信データ異常など、様々なエラーに関する対処方法を説明する。	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

「25. 組み込み開発環境に関する知識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
25. 組み込み開発環境に関するスキル	<組み込み開発の流れと環境>	<組み込み開発環境の概要>	<組み込み開発環境を用いた開発手順>	<プログラムデバッグの環境>	<デバッガソフトを使用したデバッグ環境>	<ICEを使用したデバッグ環境>	<ツールチェーンによるデバッグ>	<組み込みアプリケーションデバッグの手順>	<組み込みアプリケーションの事例と開発環境>	<組み込みクロス開発環境の構築>	<GNU開発環境の特徴>	<GNU開発環境における組み込みアプリケーション開発>	<組み込みLinux開発最新デバッグ環境>	<組み込み開発環境の評価>	<組み込み開発環境におけるデバッグのパターン演習>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_25.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
組織関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	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	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	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-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	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 [25-1, 2, 3, 4] [25-10]	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 [25-1, 4] [25-9]	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート [25-5] [25-10]	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 [25-1]	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、組み込み Linux 環境における開発支援ツールの利用手法がある。GNU 開発環境などを Linux 上で実践的に利用してデバッグを行う手法を習得する。

科目名	第9回	第10回	第11回	第12回	第13回	第14回	第15回
25.組み込み開発環境に関する知識Ⅱ	(1)組み込みアプリケーション開発の事例 (2)クリティカルな処理の実現	(1)クロス開発環境とは (2)クロス開発ツールの種類 (3)クロス開発環境の使用方法	(1)GNU 開発環境とは (2)GNU 開発環境の構成 (3)Newlib	(1)GNU 開発環境の入手 (2) GDB スタブのROM への書き込み (3)アプリケーションのコーディング、ビルド、デバッグ	(1)CPU 別の組み込みLinuxベンダ最新の対応状況	(1)開発環境の評価 (2)使用目的の明確化 (3)開発環境の評価項目 (4)開発環境の開発効率および品質向上評価	(1)プログラムデバッグ演習

（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
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-1. 組み込みアプリケーション開発事例	
対応する コースウェア	第 9 回（組み込みシステムの開発方法）	

II-25-1. 組み込みアプリケーション開発事例

組み込みアプリケーション開発の事例として、キャッシュと ROM 化の取り扱い、マイコン周辺機能、周辺 I/O 制御プログラミング、DRAM 制御プログラミングといった具体的な事例を示し、またそれぞれについての仕様や開発環境、開発の実際について解説する。

【学習の要点】

＊ 組み込みシステムのソフトウェアは、組み込みアプリケーション、組み込みミドルウェア、OS、デバイスドライバの各階層から構成される。

＊ 組み込みアプリケーションは最上位のソフトウェアで、タスクスケジューリング機能で実行されるか、割り込み機能として実行される。

＊ 組み込みミドルウェアとは、組み込みアプリケーションの機能に共通するものを集めたソフトウェアで、組み込みハードウェアベンダから提供されることが多い。

＊ 最近の傾向では、デバイスドライバは組み込みハードウェアベンダから提供されることが多いが、自社開発のハードウェアの場合は独自開発を行う必要がある。

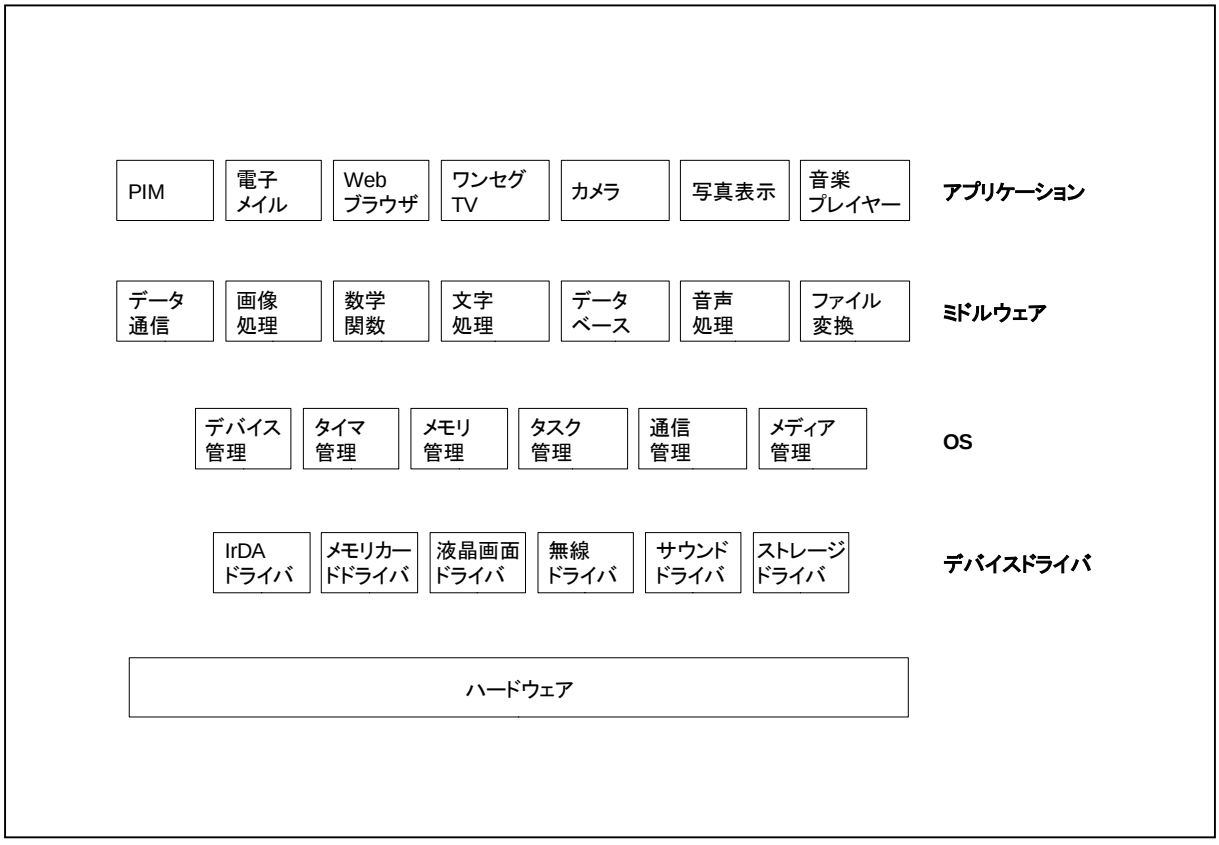


図 II-25-1.　携帯電話でのソフトウェア構成例

【解説】

1) 組み込みアプリケーションとは

組み込みアプリケーションとは、組み込みシステムの最上位に位置するソフトウェアで、ユーザや外部とインタラクションを行い、主要な機能を実現するための主要なプログラムである。

＊ 組み込みアプリケーションはタスク単位で、OS やミドルウェアのタスクスケジューリング機能で実行されるか、割り込み機能として実行される。

＊ 最近の携帯電話を例にとると、組み込みアプリケーションの種類は以下のようなものがある。

- Web ブラウザアプリケーション
- ワンセグ TV アプリケーション
- カメラ
- 音楽再生プレイヤー
- PIM(Personal Information Management)アプリケーション（電話帳、スケジュール）
- 画像ブラウザアプリケーション
- ファイルブラウザ
- バーコードリーダー

＊ 複雑な組み込みアプリケーションの開発は、ミドルウェアや OS の API 経由で行われる場合が多い。

2) 組み込みミドルウェアとは

組み込みミドルウェアとは、組み込みアプリケーションの機能に共通するものを集めたソフトウェアである。組み込みアプリケーションはミドルウェアをベースに作り上げられている。

- 通信プロトコル： Web ブラウザやメイルアプリケーションで利用される。
- ファイルシステム： ファイルブラウザで利用される。
- 画像認識/音声認識：バーコードリーダーで利用される。
- 数学ライブラリ：浮動小数点計算を始めとして、数学計算で利用される。
- 周辺 I/O 制御：メモリカードの読み込みや、赤外線通信等で利用される。
- DRAM 制御：ゲーム等のメモリが大きく必要なアプリケーションで利用される。

＊ 組み込みミドルウェアは、組み込みハードウェアベンダや組み込み開発環境、または OS の機能として提供されることが多い。

3) OS/デバイスドライバ

＊ 最近は組み込みシステムにおいて Linux, Windows CE 等の OS を利用することが多い。詳細については、(II-25-8 組み込み Linux システムの開発環境)を参照のこと。

＊ デバイスドライバはハードウェアに直接アクセスし、制御するソフトウェアである。一昔前の組み込みアプリケーション開発の大部分は、デバイスドライバ開発であった。

＊ 最近の傾向では、デバイスドライバは組み込みハードウェアベンダから提供されることが多いが、自社開発のハードウェアの場合は独自開発を行う必要がある。

- デバイスドライバ開発にはハードウェアとの通信の仕組みを把握すると共に、ターゲットプロセッサについてのタイマ機能、ポーリングや割り込み、I/O 制御などの正確な知識が必要である。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-2. 組み込みアプリケーション開発における留意点	
対応する コースウェア	第 9 回（組み込みシステムの開発方法）	

II-25-2. 組み込みアプリケーション開発における留意点

スタートアップルーチンの実装や例外処理、安全化設計、ROM 化において気をつけるべき点、低消費電力モード、Watch Dog Timer (WDT)の利用など、組み込みアプリケーション開発において留意すべき様々な事項を紹介する。

【学習の要点】

- スタートアップルーチンはスタックの確保に注意する。
- 例外処理はプログラムが肥大化することがある。
- ROM 化は手間がかかるので、ROM ライタを利用する前に十分にテストをする。
- 安全化設計では、バグが起きても被害を最小にするように設計する。
- ウォッチドッグタイマは異常が起きていないかを監視する目的で使用されるタイマで、暴走や異常処理を検知する。

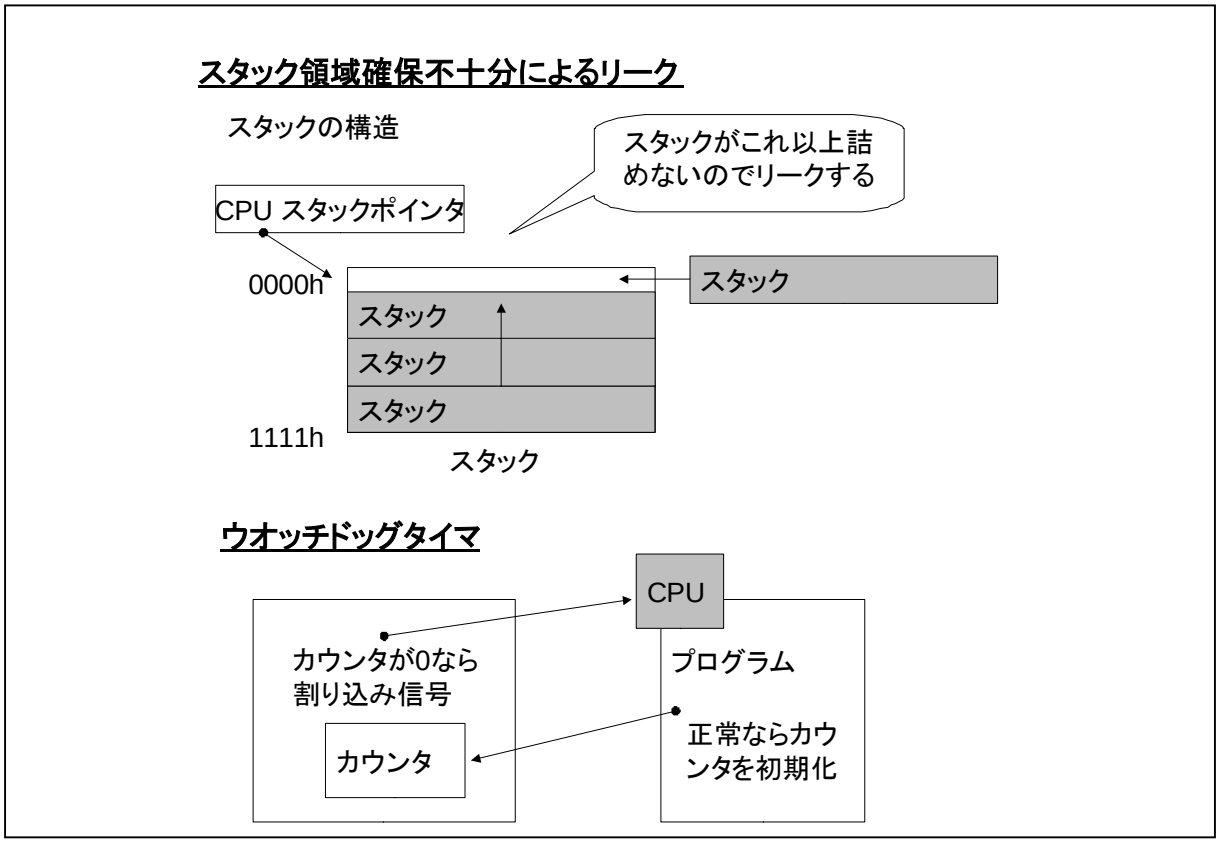


図 II-25-2. スタック領域確保不十分によるリークとウォッチドッグタイマ

【解説】

1) スタートアップルーチン

＊ コンピュータは電源が投入される、またリセットをされたときにブートとよばれる起動動作を行う。ことのときに最初に呼び出され、実行されるプログラムがスタートアップルーチンである。

＊ スタートアップルーチンは ROM にあり、起動されると以下の動作を順に行う。

- CPU のレジスタを設定し、ハードウェア・ソフトウェアを初期化する。
- ソフトウェアを初期化し、プログラムが動作するために必要なヒープとスタックメモリを設定した後に、割り込み関連の設定を行う。
- OS が存在する場合は、まず OS を起動する。OS が存在しない場合はアプリケーションのメインルーチンにジャンプする。

＊ スタートアップルーチンの実装において気をつけるべきことは、スタックメモリの設定である。スタック領域の確保が十分ではないと、ヒープがスタックを浸食しデータを破壊するという事態になる。

2) 例外処理

＊ 一般的に例外処理をもつ言語系は、例外処理とデストラクタが一組になるように実装されている。これは、コンパイラにおいて生成されるコードが、例外処理が送出されるタイミングでデストラクタを実行するようになっていることである。このコードがプログラムの肥大化を招く事態になる。

＊ また例外が適切にコーディングされない場合、メモリインスタンスのリークが起きる場合がある。これを例外安全性の破綻という。

3) ROM 化

＊ 組み込みにおいては、スタートアップルーチンなど電源投入により起動するプログラムは ROM 化する必要がある。このときに ROM ライタを利用するが、この作業は手間がかかるので、手戻りがないように十分にテストを行ってから ROM 化をする必要がある。

4) 安全化設計

＊ 特にミッションクリティカルな場面で組み込みシステムが使用される場合、バグが発生すると大きな事故や損害を招く事態になる可能性がある。これを防ぐ設計手法を安全化設計という。

＊ 安全化設計の基本は、アプリケーションのバグが発生してもシステム全体として、被害が最小になるようにとどめることである。これを実現するためには次の指針が必要である。

- リスクの軽重により、モジュールを分離する。
- シンプルな設計を採用し、バグの混入、例外の発生を抑える。

5) ウォッチドッグタイマ

＊ ウォッチドッグタイマはシステムから独立して暴走や異常処理が発生していないかを監視し検知する目的で使用されるタイマである。

＊ ウォッチドッグタイマの内部にはカウンタを持っていて、周期的にカウンタをカウントダウンすることで、カウンタ値が 0 になったらタイマ割り込みを発生させる。プログラムや OS は周期的にウォッチドッグタイマのカウンタを初期化することで、正常であることをウォッチドッグタイマに伝達する。正常ならば、カウンタが初期化し続けられ、タイマ割り込みが発生しないということになる。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-3. クロス開発環境の種類と特徴	
対応する コースウェア	第 10 回(組み込みクロス開発環境の構築)	

II-25-3. クロス開発環境の種類と特徴

組み込みアプリケーションを開発するためのクロス開発環境についてその全体像と構成、特徴、必要性等を説明する。さらにクロス開発ツールの種類として半導体ベンダが提供するもの、サードパーティが提供するもの、Free Software Foundation の提供による GNU 開発環境などがあることを挙げ、それぞれを簡単に紹介する。

【学習の要点】

* 開発する環境と実行する環境が違う状態で行う開発方法をクロス開発という。組み込みシステム開発では、クロス開発を利用して開発を行うことがほとんどである。
* クロス開発においてプログラム開発を行う環境をホスト環境といい、開発したプログラムを実行させる環境を実行環境という。
* クロス開発環境の種類には、半導体ベンダ提供するもの、クロス開発環境ベンダが提供するもの、そして Free Software Foundation(FSF)が提供するものがある。

種類	製品/ツール名	ベンダー名	対応プロセッサ
半導体ベンダー	RealView Developer Suite	ARM	ARM
	High-performance Embedded Workshop	ルネサステクノロジ	H8, SH
	FR Family SOFTUNE	富士通	FR, FR-V
	SP850	NEC	V850
	CodeWarrior Development Tools	Freescale Semiconductor	M68K, HC(S)08/RS08, ColdFire, HCS12(X), MPC500, 56800/E, 他
クロス開発環境ベンダー	GNUPro	Red Hat	ARM, calmrisc, cy16, d10v, d30v, fr30, frv, h8300, i386, i960, ip2k, ip4k, iq2000, m32c, M68k, mcore, mep, mips (all ISAs), mn10200, mn10300, ms1, sh (32/64), SPARC, sparclite, V850, vpe2k, vpe4k, xscale(arm), xstormy16
	exeGCC	京都マイクロコンピュータ	ARM、SH、MIPS、V850E、MN103
	GNUWing	Upwind Technology	ARM, PowerPC, MIPS, SH
Free Software Foundation(FSF)	binutils, gcc, newlib, gdb		ARM, x86, MIPS32, PowerPC, SH, H8/300H, M32R, V850, FR, FR-V

図 II-25-3.　主なクロス開発環境

【解説】

1) クロス開発とは

＊ クロス開発とは開発する環境と実行する環境が違う状態で行う開発方法のことである。なお、開発環境と実行環境が同じプログラム開発方法をセルフ開発という。

＊ 組み込みシステムを開発する場合、ターゲットとなる組み込み機器は多様であり、ターゲットと開発環境を同じにすることが現実的ではない。したがって、組み込みシステム開発では、クロス開発環境を利用して開発を行うことがほとんどである。

＊ クロス開発においてプログラム開発を行う環境をホスト環境といい、開発したプログラムを実行させる環境を実行環境という。

＊ ターゲット環境での動作環境が完了したプログラムは、スタートアップルーチンや OS を含めてファームウェアという形でまとめられ、ROM 化されたり、不揮発性メモリに焼き込まれたりする。

＊ クロス開発環境で中心となるのは、ツールチェーンと呼ばれる一連の開発キットである。ツールチェーンにはバイナリユーティリティ、C コンパイラ、C ライブラリ等が含まれる。

2) 主なクロス開発環境

＊ クロス開発環境の種類には、半導体ベンダ提供するもの、クロス開発環境ベンダが提供するもの、そして Free Software Foundation(FSF)が提供する 3 つがある。

＊ 半導体ベンダが提供するもの

- RealView Developer Suite （ARM） 対応プロセッサ：ARM
- High-performance Embedded Workshop（ルネサステクノロジ） 対応プロセッサ：H8, SuperH
- FR Family SOFTUNE（富士通） 対応プロセッサ：FR, FR-V 向け
- SP850(NEC) 対応プロセッサ：V850
- CodeWarrior Development Tools (Freescale Semiconductor)
  対応プロセッサ：M68K, HC(S)08/RS08, ColdFire, HCS12(X), MPC500, 56800/E, 他

＊ クロス開発環境ベンダが提供するもの

- GNUPro (Red Hat)
  対応プロセッサ：ARM, calmrisc, cy16, d10v, d30v, fr30, frv, h8300, i386, i960, ip2k, ip4k, iq2000, m32c, M68k, mcore, mep, mips (all ISAs), mn10200, mn10300, ms1, sh (32/64), SPARC, sparclite, V850, vpe2k, vpe4k, xscale(arm), and xstormy16
- exeGCC（京都マイクロコンピュータ）
  対応プロセッサ： ARM、SH、MIPS、V850E、MN103
- GNUWing (Upwind Technology)
  対応プロセッサ：ARM, PowerPC, MIPS, SH

＊ Free Software Foundation(FSF)の提供するもの

- binutils, gcc, newlib, gdb
  対応プロセッサ：ARM, x86, MIPS32, PowerPC, SH, H8/300H, M32R, V850, FR, FR-V

＊ どのクロス開発環境を選択するかは、ターゲットとなる組み込み機器の CPU ベンダによるクロス開発環境を入手できるか、組み込み OS の有無・種類などを十分に検討して選択する必要がある。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-4. クロス開発環境を利用した組み込みシステム開発	
対応する コースウェア	第 10 回(組み込みクロス開発環境の構築)	

II-25-4. クロス開発環境を利用した組み込みシステム開発

クロス開発環境を利用した組み込みアプリケーションの開発について、開発環境の選択方法や開発環境の構築、クロス開発環境を活用した実際のアプリケーション開発手順等を解説する。さらに複数のターゲットを対象としたクロス開発やライセンスの取り扱いなど、関連する事項についても触れる。

【学習の要点】

* ホスト環境とターゲット環境に合致したクロス開発ツールチェーンを入手し、適切にセットアップを行わないと、ホスト環境でビルドができるがターゲット環境で動作しないというロードモジュールができてしまう。
* 組み込みソフトウェアにおいて、ソフトウェアの既存モジュールの利用に関しては、ライセンスに対して十分に検討する必要がある。
* OSS ソフトウェアの組み込みへの利用にあたっては、ソースコードの公開など、ライセンスに従った対応が求められる。

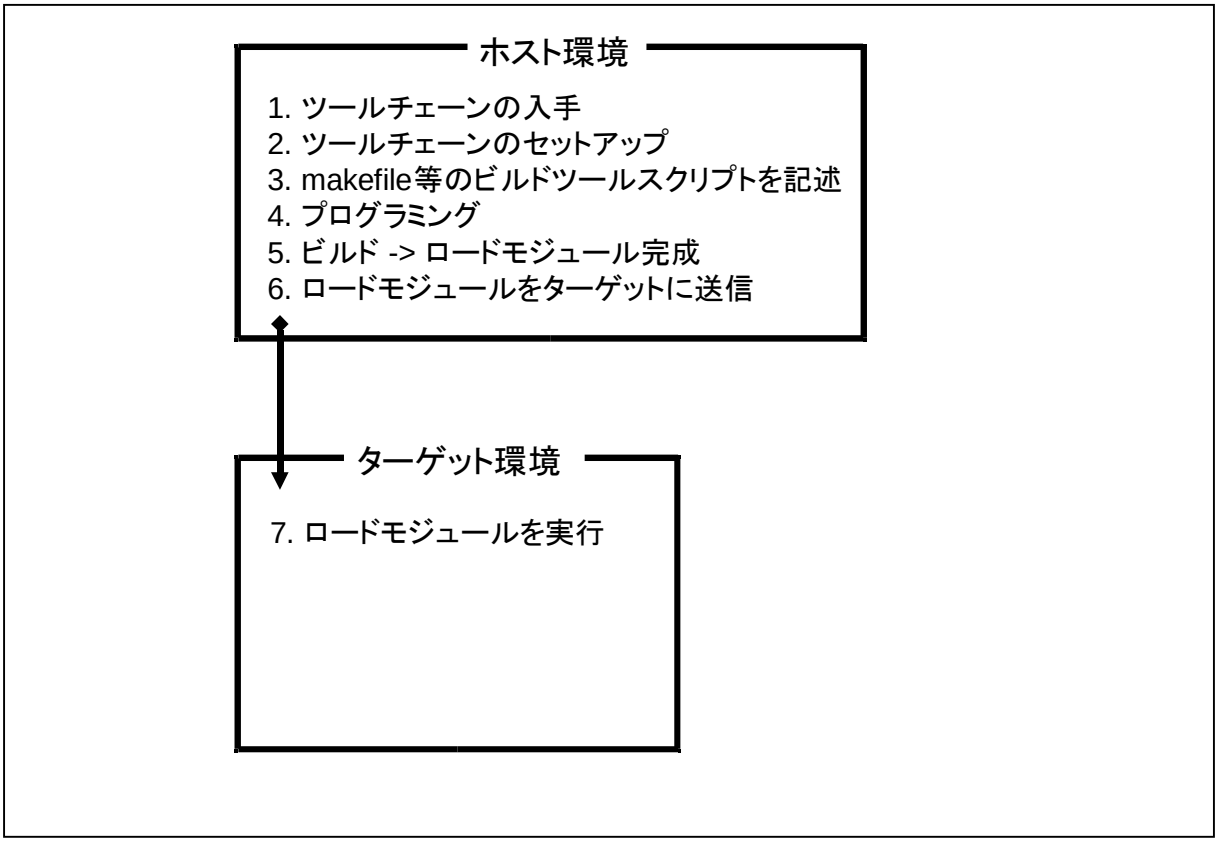


図 II-25-4.　クロス開発環境での開発手順

【解説】

1) クロス開発における組み込みシステム開発

＊ 組み込みシステム開発をクロス開発環境で行うための準備は次の通りである。

- ホスト環境とターゲット環境に合致したクロス開発ツールチェーンを入手する。
- ホスト環境にツールチェーンをビルドまたはセットアップする。注意すべきことは、開発言語となる言語のコンパイラとライブラリを適切なファイルパスにセットアップしなければならないことである。
- ツールチェーンを利用するために必要な Makefile、Ant、または Maven 等のビルドツールスクリプトを記述する。この際にターゲット環境のコンパイラ、リンカを適切に指定しないと全く動作しないロードモジュールができてしまう。

＊ クロス開発環境で開発を行い実行するまでの手順は次の通りである。

- クロス開発環境がセットアップできたら、プログラミングを行う。
- プログラムが完成したらビルドを行い、ターゲット環境のロードモジュールを作成する。
- シリアル、Ethernet 等の通信を利用して、ロードモジュールをターゲットシステムに送り込む。このときに、NFS 等のファイル共有を利用すると手間が省けるときがある。
- ロードモジュールを実行する。

2) ライセンス

＊ 機能要件があまりに膨大になっている組み込みシステム開発の現状においては、既存の定評あるソフトウェアモジュールを利用することが理に叶っている場合が多い。その場合、ライセンスに対して十分に検討し、知的財産権（IP）の不正使用を起こさないように心がける必要がある。

＊ ライブラリやソフトウェアの既存モジュールのライセンス種類は、大きく分けてつぎの二つがあるが、どちらも掲げているライセンスに対して十分に検討し、IP の不正使用を起こさないように心がけなければならない。

- 商用ソフトウェアモジュールの利用
- OSS の利用

＊ OSS の利用にあたっては、その OSS が採用するライセンス条項を確認し、対応することが求められる。一例として、GPL（詳細は II-2-1 参照）である Linux を採用した携帯電話では、メーカの Web サイトでその製品の利用者に対してソースコードを公開する対応をとっている。ソースコードの公開方法は、その製品のユーザに対して実施することでライセンス上はクリアできるため、製品のシリアルコードを入力させる方法をとっている場合もある。

＊ ライセンスに違反した場合、企業イメージの低下につながるリスクがある。（詳細は II-2-9 参照）

＊ FSF（Free Software Foundation）は、GPL ライセンスを採用した OSS の利用における新たな課題として「Tivoization」を提起している。これは米国を中心に販売されている Linux ベースのデジタル・レコーダ「TiVo」において、ユーザが改変したプログラムの実行を機器側で制限しており、ユーザが購入した製品を「自由」に使うことを妨げている、というものである。FSF で策定した GPL の最新版である GPLv3 では、Tivoization を防ぐ内容が盛り込まれている。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-5. GNU 開発環境の利用方法	
対応する コースウェア	第 11 回（GNU 開発環境の特徴）	

II-25-5. GNU 開発環境の利用方法

GNU 開発環境の全体像と構成、特徴、必要性について説明する。また GNU 開発環境を用いた開発の手順や、それ以外の開発環境との比較、GNU 開発環境を利用した際におけるライセンスの取り扱いやサポート等、関連するトピックについても触れる。

【学習の要点】

* 組み込み OS として Linux を選択する場合、高品質と信頼性、幅広いハードウェアサポート等の理由から、GNU 開発環境を用いる場合が多い。
* GPL(GNU Public License)で提供されているライブラリやモジュールを静的リンクしてロードモジュールを作成した場合、そのロードもジュールは GPL の派生物となってしまう。
* GCC 開発環境を利用して、組み込みアプリケーションのクロスビルドを行うのためには、ターゲット環境用の gcc を利用し、リンクにはターゲット環境用の ld を利用する。ビルド、リンクのオプションは正しく使うようにする。
* makefile は一度作成すると、手動で gcc, ld を実行する手間を省け、効率のよい開発を実現する。

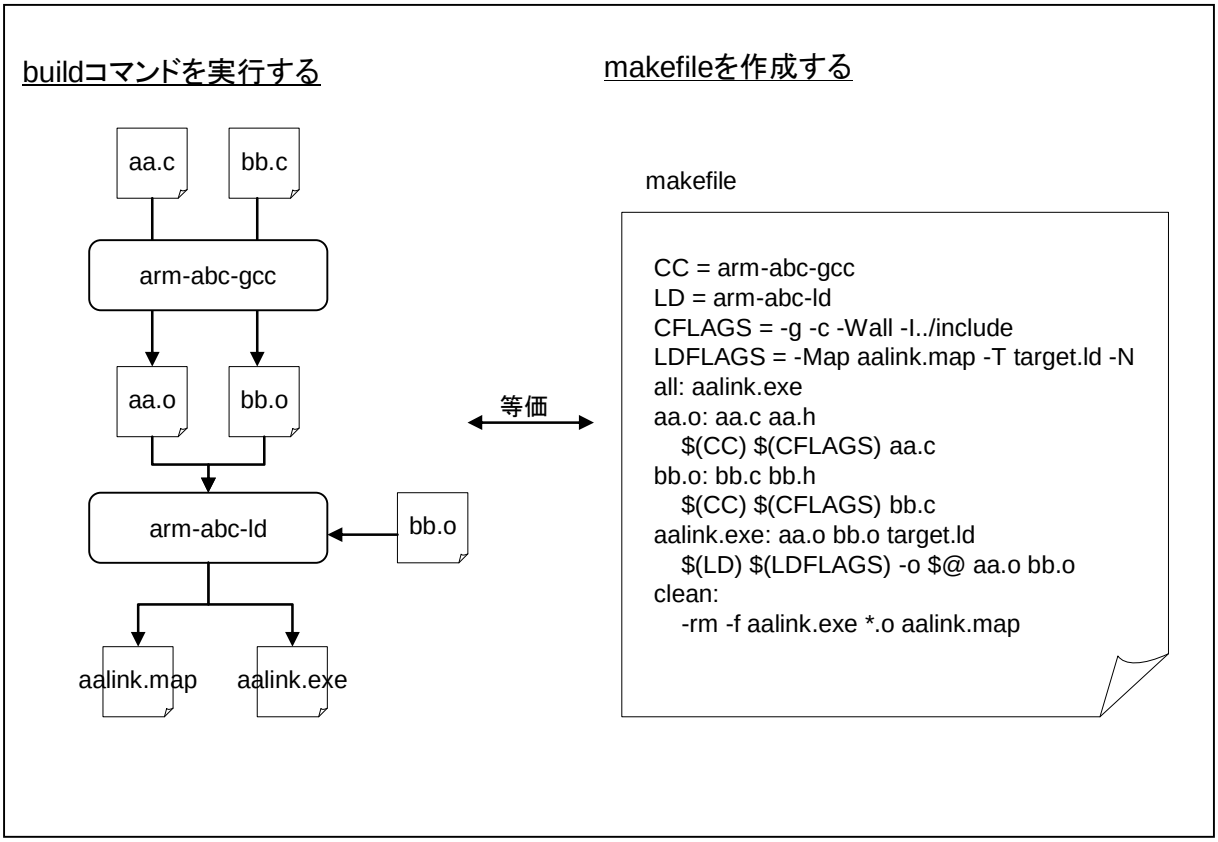


図 II-25-5.　GNU 開発環境の利用方法

【解説】

1) GNU 開発環境

＊ 組み込み OS として Linux を選択する場合、現在では GNU 開発環境を用いる場合が多い。

＊ GNU 開発環境を利用する優位性は、コードの入手可能性、高品質と信頼性、幅広いハードウェアサポート、通信スタック等のソフトウェアスタックの充実、コミュニティからの支援、ベンダからの独立、無料等があげられる。

＊ GNU 開発環境の入手は ftp://ftp.gnu.org/gnu より入手する。

＊ GNU 開発環境は、著作権や再配布についてのライセンス条項に気をつける必要がある。特に GPL(GNU Public License)で提供されているライブラリやモジュールを静的リンクしてロードモジュールを作成した場合、そのロードモジュールは GPL の派生物となってしまうことには、注意を要する。

＊ GNU 開発環境以外に、Eclipse や KDevelop などの開発環境を利用することもできるが、その違いについては十分に調査する必要がある。

2) コンパイルからリンクまで

＊ 前提条件

- aa.c, bb.c というソースファイルがあり、aa.c は bb.c の関数を呼び出しているとする。それぞれのヘッダファイルは/include フォルダにある。ターゲットのリンカスクリプト target.ld とする。ターゲット名は arm-abc とする。従って C コンパイラは arm-abc-gcc であり、リンカは arm-abc-ld となる。

＊ コンパイルコマンド:

```
$ arm-abc-gcc -g -c -Wall -I../include aa.c
$ arm-abc-gcc -g -c -Wall -I../include bb.c
```

＊ -g オプションはデバッグ情報の生成のためで、-c オプションはコンパイルのみ行いリンクを抑制するために用いた。-Wall オプションはすべての警告を出力するためで、-I オプションはヘッダファイルを格納しているディレクトリのサーチパスを指定する。

＊ コンパイルにより生成されたのは、aa.o, bb.o

＊ リンクコマンド:

```
$ arm-abc-ld -Map aalink.map -T target.ld -N -o aalink.exe aa.o bb.o
```

＊ -N オプションは生成されたロードモジュールのテキストセクション、データセクションを読み出し/書き出し可能にする。

＊ リンクにより生成されたのは、aalink.exe, aalink.map

＊ aalink.exe はロードモジュールでこれをターゲット環境に転送して実行する。aalink.map はコードとデータのアドレスのマッピングファイルでデバッグのために用いる。

3) makefile

＊ 前節と等価な操作を makefile を作成することで実現できる。

＊ makefile を作成したら以下のように実行する。

```
$ make
```

＊ 生成したファイルを削除したいときは次のようにする。

```
$make clean
```

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-6. GNU 開発環境を構成するツール群	
対応する コースウェア	第 11 回 (GNU 開発環境の特徴)	

II-25-6. GNU 開発環境を構成するツール群

GCC や GDB といった基本的なツールに加え、as (アセンブラ)、ld (リンカ)、nm (シンボルテーブルの表示)、objdump や objcopy といったオブジェクトファイルを操作するツール、オブジェクトファイルからシンボルを削除する strip など、binutils (バイナリユーティリティ)に含まれる各種のツールを説明する。

【学習の要点】

* GNU 開発環境における代表的な C コンパイラは gcc であり、C ライブラリは glibc である。
* バイナリユーティリティパッケージは binutils である。
* デバッギングを行うには gdb を利用することがほとんどである。
* GNU の開発ツール群のファイル名のプレフィックス部分には、ターゲット環境名を用いることが慣習として行われている。
* ホスト、ターゲット、そしてカーネルの組み合わせにより、必要な binutils, gcc, glibc のバージョンの組み合わせ、パッチ当ての有無が変わってくるので注意を要する。

ツール	ツール名	名前例	場所
コンパイラ	gcc g++ c++	arm-linux-gnueabi-gcc arm-linux-gnueabi-g++ arm-linux-gnueabi-c++	{toolchain}/bin
ライブラリ	glic uClibc	libc.so.6	{toolchain}/lib インストールした場所
リンカ	ld	arm-linux-gnueabi-ld	{toolchain}/bin
アセンブラ	gas	arm-linux-gnueabi-gas	{toolchain}/bin
アーカイブ ユーティリティ	ar	arm-linux-gnueabi-ar	{toolchain}/bin
シンボル表示 ユーティリティ	nm	arm-linux-gnueabi-nm	{toolchain}/bin
デバガ	gdb	arm-linux-gnueabi-gdb	{toolchain}/bin
リモートデバガ	gdbserver	arm-linux-gnueabi-gdbserver	{toolchain}/bin

図 II-25-6.　GNU 開発環境を構成するツール群

【解説】

1) GNU 開発環境コンパイラ

* GNU 開発環境における代表的な C コンパイラは gcc である。組み込み開発環境ではそれほど利用されてはいないが、C++のコンパイラとして c++や g++が利用されることがある。ツールチェーンとして提供されるコンパイラはツールチェーンホームディレクトリの下の/bin ディレクトリにある場合が多い。
* コンパイラはターゲット環境で動作するものを十分に検討して選択しなければならない。通常コンパイラのファイル名はプレフィックスでターゲット環境の CPU を示している。例えば、ARM を CPU に利用している場合だと次のような名前となる。
  arm -linux-gnueabi-gcc

2) GNU 開発環境ライブラリ

* GNU 開発環境における代表的な C ライブラリは glibc である。たいていはツールチェーンの下にある lib ディレクトリに、libc-x.x.x.so というダイナミックリンクライブラリの形で提供されていることが多い。glibc のシンボリックリンクは libc.so.x になる。
* C ライブラリとして glibc の代わりに uClibc （Ⅱ-27-6 参照)を利用することもできる。

3) GNU 開発環境バイナリユーティリティパッケージ

* GNU 開発環境におけるバイナリユーティリティパッケージは binutils である。binutils には、ld, gas, ar, nm が含まれる。ld はリンカ、gas はアセンブラ、nm はシンボルテーブル表示ユーティリティである。
* objdump はオブジェクトファイルのデバッグ情報やヘッダ・セクション情報を表示するツールである。objcopy はオブジェクトファイルのコピーや raw バイナリファイルに変換するツールである。
* strip は、ライブラリからシンボルテーブルを取り除いてオブジェクトサイズを削減するユーティリティである。組み込みシステムにはメモリサイズの強い制限があるので、このツールを利用して通常の OS で使用しているライブラリを組み込み用に変換する。
* binutils のバイナリユーティリティはたいていツールチェーンの bin ディレクトリに、コンパイラと共に提供されることが多い。

4) GNU デバッグ環境

* GNU 開発環境でデバッギングを行うには gdb を利用することがほとんどである。gdb についてもターゲット環境に相当するものを利用しなければならない。たいていの場合、ファイル名のプレフィックス部分にターゲット環境名が入っている。
  arm-linux-gdb
* gdb を利用してシンボリックデバッギングを行うには、コンパイル時に-g オプションが必要である。
* gdb には gdbserver というリモートデバッギングを可能にするモジュールがある。

5) ホスト、ターゲット、カーネル

* ホスト、ターゲット、そしてカーネルの組み合わせにより、必要な binutils, gcc, glibc のバージョンの組み合わせ、パッチ当ての有無が変わってくるので注意を要する。この構成管理が、長く開発を続けるコツになる。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-7. GNU 開発環境を用いたアプリケーション開発	
対応する コースウェア	第 12 回 (GNU 開発環境における組み込みアプリケーション開発)	

II-25-7. GNU 開発環境を用いたアプリケーション開発

GNU 開発環境による組み込みアプリケーション開発の手順について、その特徴と役割を概説する。また GNU 開発環境の入手方法、環境構築の方法、GDB スタブの ROM 書き込み、アプリケーションのコーディングとビルドおよびデバッグといった具体的な手順の内容を説明する。

【学習の要点】

* GNU クロス環境においてデバッグを行うには、ターゲットマシンにおいて通常の方法で gdb を実行、gdbserver でリモートデバッギング、ターゲットに組み込みのデバッグモニタを使用、という三つの方法がある。
* ターゲットマシンにおいて通常の方法で gdb を実行する方法は、ターゲット側のリソースが豊かでないと実現できない。
* gdbserver を利用する方法は、ホストとターゲットの間をシリアルまたは Ethernet に繋がれた通信環境が必要である。ロードモジュールは双方に配置する。
* デバッグモニタを使用する場合は、予めターゲットにデバッグモニタが組み込まれている必要がある。

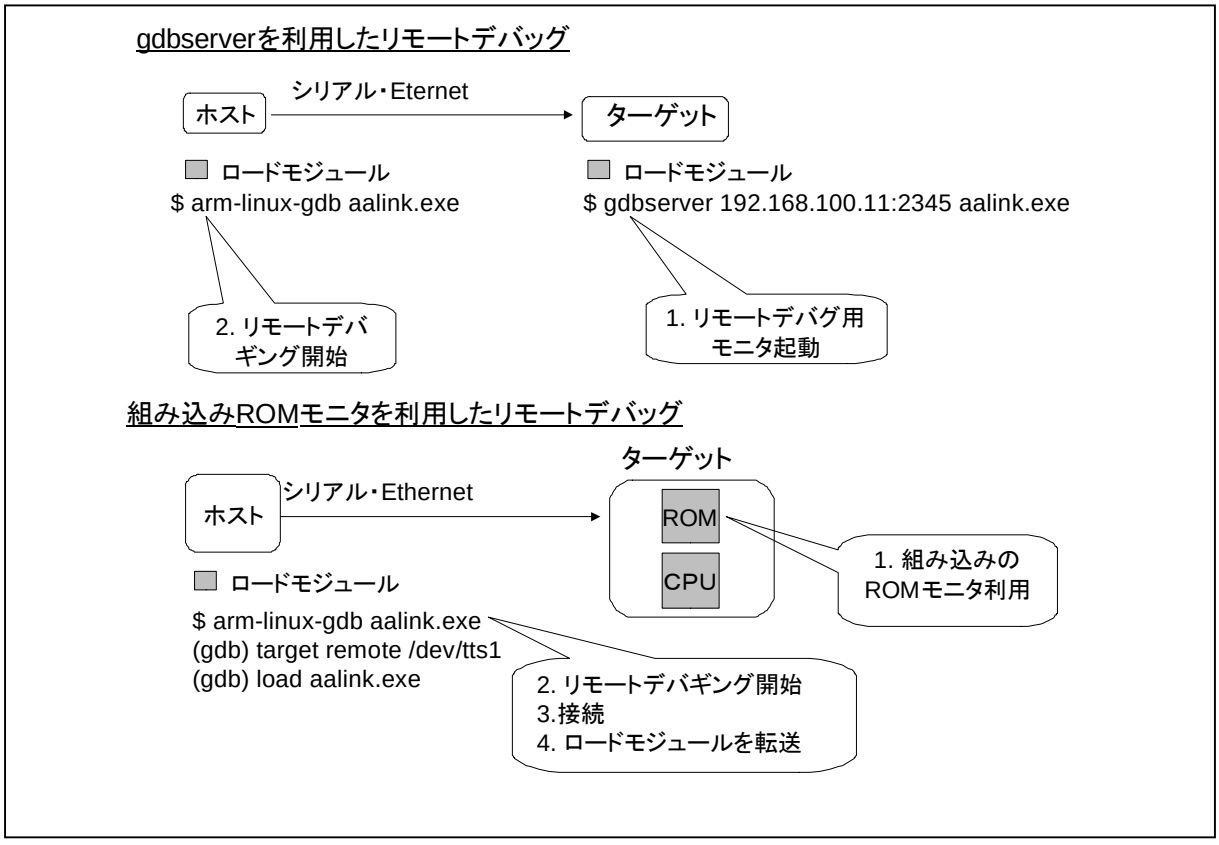


図 II-25-7. リモートデバッグ

【解説】

1) アプリケーション開発の概要

＊ GNU 開発環境の入手は ftp://ftp.gnu.org/gnu より入手する。

＊ 環境構築について、ホスト、ターゲット、そしてカーネルの組み合わせにより、必要な binutils, gcc, glibc のバージョンの組み合わせ、パッチ当ての有無が変わってくるので注意を要する。

2) GNU クロス環境におけるデバッギング

＊ GNU クロス環境においてデバッギングを行うには次の三つの方法がある。a)ターゲットマシンにおいて通常の方法で gdb を実行、b)gdbserver でリモートデバッギング、c)ターゲットに組み込みのデバッグモニタを使用

3) ターゲットマシンにおいて通常の方法で gdb を実行

＊ この方法はターゲット環境に最小限のビルド環境を構築する必要があり、ターゲットのスペックが相当高くないと利用できない。gdb は相当のヒープメモリを消費するからである。

＊ この方法は通常の gdb デバッギングなので、組み込みであることを特に意識しなくてよい。

4) gdbserver でリモートデバッギング

＊ 前提:ホストとターゲットは同じセグメント内にある Ethernet で接続されていて、2345 ポートが空いている。ホストの IP は 192.168.100.11 とする。さらにターゲットもホストも Linux が動作している。ロードモジュールファイル名は aalink.exe で、-g オプションでコンパイル済みである。

＊ ターゲット側の gdbserver を起動する。

```
$ gdbserver 192.168.100.11:2345 aalink.exe
```

＊ ホスト側にもターゲットと同様のロードモジュールを配置する。

＊ ホスト側の gdb を実行してデバッギングを開始する。

```
$ arm-linux-gdb aalink.exe
(gdb) target remote 192.168.100.12:2345
Remote debugging using 192.168.100.12:2345
0x10000061 in _start()
```

＊ gdbserver ではターゲット側でプログラムが終了しても、もう一度ターゲット側の gdbserver を立ち上げ直すだけで、ホスト側の gdb を再起動する必要はない。

＊ gdbserver は Ethernet だけでなく、シリアルポートにも対応している。

5) ターゲットに組み込みのデバッグモニタを使用

＊ 前提:ターゲットには付属のデバッグモニタ DebMon があり、DebMon はシリアルポート ttys1 を通じてホストと通信する。ホスト側に-g オプションでコンパイル済みロードモジュール aalink.exe が配置してある。

＊ ホスト側の gdb を起動する。 $ arm-linux-gdb aalink.exe

＊ ホスト側からターゲットのデバッグモニタのシリアルポートに繋ぐ

```
(gdb) target remote /dev/tts1
Remote debugging using /dev/ttys1
```

＊ ロードモジュールをターゲット上にダウンロードしてデバッグを開始する。

```
(gdb) load aalink.exe
```

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-8 組み込み Linux の開発環境	
対応する コースウェア	第 13 回（組み込み Linux 開発最新デバッグ環境）	

II-25-8 組み込み Linux の開発環境

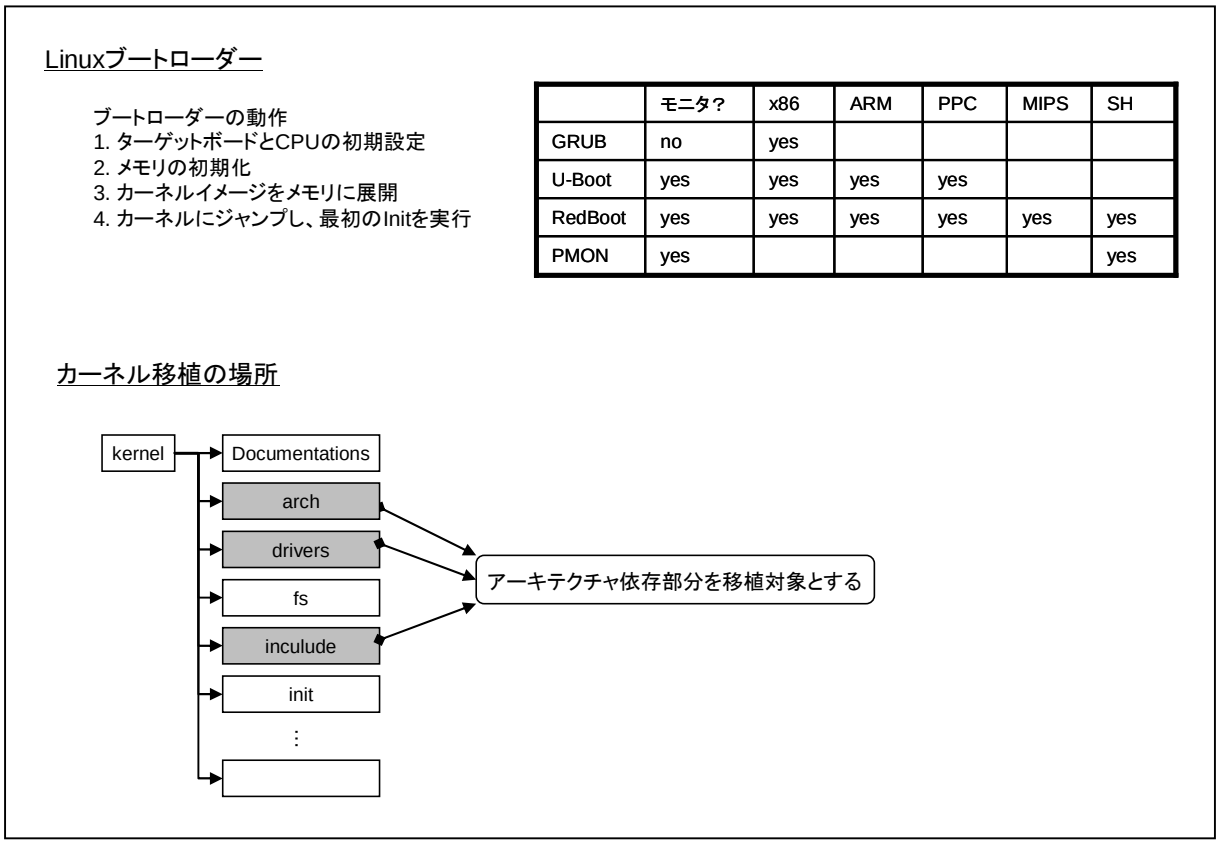
組み込み Linux を用いた開発環境について説明する。その全体像と構成、組み込み Linux 開発環境を利用する必要性を解説する。また ARM、PXA、SH など、CPU の系列別に組み込み Linux ベンダの対応状況を紹介する。

【学習の要点】

＊ 組み込みシステムの OS として Linux を採用する動きは年々加速している。その理由として、コードが入手可能であることや、信頼性に富んでいるといったことに加えて、ネットワークに関するソフトウェアスタックが充実していていることがあげられる。

＊ ベンダの組み込みLinuxシステムを利用する場合は、カーネル移植とデバイスドライバ開発がないために、大幅な工数の削減が実現できる。

＊ 最近は汎用組み込み Linux ベンダよりも上位のマザーボードベンダが、移植済み、デバイスドライバ開発済みの組み込み Linux 及びツール群を提供する。



	モニタ?	x86	ARM	PPC	MIPS	SH
GRUB	no	yes				
U-Boot	yes	yes	yes	yes		
RedBoot	yes	yes	yes	yes	yes	yes
PMON	yes					yes

図 II-25-8 Linux ブートローダーの動作とカーネルの移植場所

【解説】

1) 組み込み Linux 概要

＊ 組み込みシステムの OS として Linux を採用する動きは年々加速している。特にネットワーク系の機能を必要とするインターネットアプライアンス、デジタル TV、プリンタ、ゲーム機、PDA、携帯電話等については、Linux の採用が著しい。

＊ Linux はネットワークに関するソフトウェアスタックが充実していているために、組み込みアプリケーションの開発が非常に簡単になる。

＊ Linux を組み込み OS として採用する理由は、ソースコードの入手可能性、高品質と信頼性、幅広いハードウェアサポート、通信スタック等のソフトウェアスタックの充実、コミュニティからの支援、ベンダからの独立、無料等がある。

2) 組み込み Linux ベンダ情報

＊ 組み込み Linux ベンダには、主に ARM、SH、PPC、PXA 等の CPU の系列別に分けられるが、ベンダの参入撤退が激しく、情報の新陳代謝が盛んである分野なので、Web 等でその対応状況を確認する必要がある。

＊ 組み込み Linux ベンダの状況が網羅的に把握することできる Web サイトとして、2008 年 9 月現在では、@IT MONOist(http://monoist.atmarkit.co.jp/fembedded/link/link_linux_b.html)がある。

3) 組み込み Linux のブートとブートローダー

＊ Linux は自身をブートする仕組みを持っていない。従って Linux カーネルを起動するために必要な以下の動作については、ブートローダーが必要になる。

- ターゲットボードと CPU の初期設定
- メモリの初期化
- カーネルイメージをメモリに展開
- カーネルにジャンプし、最初の Init を実行

＊ 組み込みマザーボードメーカーの物を利用するのならば、そのファームウェアにブートローダーが含まれる場合がほとんどである。

＊ オープンソースのブートローダーという選択肢もある。その場合対象としているアーキテクチャに気をつける必要がある。以下は主なブートローダーの対応アーキテクチャである。
GRUB(x86), U-Boot(x86, PPC, ARM), RedBoot(x86, ARM, PPC, MIPS, SH), PMON(MIPS)

4) Linux カーネルのターゲットボードへの移植

＊ Linux カーネルの移植は C 言語またはアセンブラで行う。

＊ Linux カーネルのソースコードは CPU アーキテクチャ依存部とそうでない部分に分けられている。このアーキテクチャ依存部に対するコーディングを行う。例えば arc/以下や include/以下、またはドライバ drivers/以下である。

＊ カーネルの移植をおこなったら、そのパッチを各アーキテクチャコミュニティに提供することが Linux 全体の持続的発展につながる。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-9. 組み込み開発環境の評価	
対応する コースウェア	第 14 回（組み込み開発環境の評価）	

II-25-9. 組み込み開発環境の評価

購入や保守にかかる費用、操作性の善し悪し、操作方法の習得にかかる期間、前提とする知識、ターゲットシステムのハードウェアとの適合性、他ツールとの接続性や互換性等、組み込みアプリケーション開発環境を評価する項目と評価方法について説明する。

【学習の要点】

＊ ベンダの組み込みLinuxシステムを利用する場合は、カーネル移植とデバイスドライバ開発がないために、大幅な工数の削減が実現できる。

＊ 最近は汎用組み込み Linux ベンダよりも上位のマザーボードベンダが、移植済み、デバイスドライバ開発済みの組み込み Linux 及びツール群を提供する。

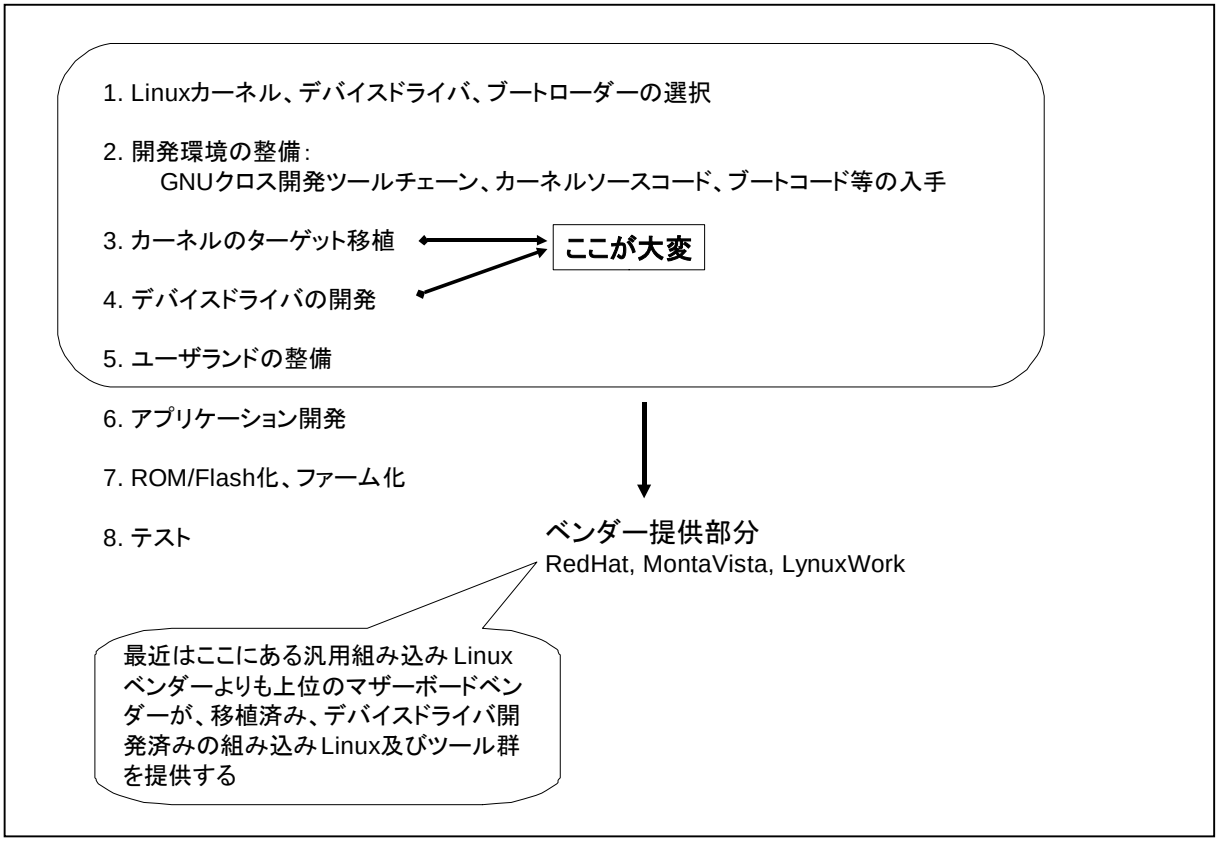


図 II-25-9. 組み込み Linux システムの開発

【解説】

1) 開発環境の評価

＊ ソフトウェア開発環境を評価するときに、一般的なポイントは、開発環境自体の機能や性能、操作性、そして価格や保守性である。

＊ 組み込みシステム独特の問題として、ターゲットとする組み込み機器がどのような種類のものであるかという点がソフトウェア開発環境の評価に必要なときもある。

＊ ROM エミュレーター(ROM-ICE)の機能を持っている開発環境が必要になることもある。

＊ LSI にオンチップ・デバッグ機能を利用する開発環境が必要になることもある。

2) 組み込み Linux 開発環境の評価

＊ 組み込みLinuxシステムにおける開発環境は主にオープンソースのコミュニティによるものと、ベンダによるものがある。ベンダは Red Hat, MontaVista, LynuxWorks などがある。それぞれに対応できる CPU が決まっているので、ベンダを選択する際は最新の情報にあたること。

＊ ベンダによる組み込み Linux システムには開発環境として、GNU 開発ツールチェーンの他に、GUI ベースの IDE(統合開発環境)やトレースツール、ソースコード解析ツールが含まれ充実していることが多い。

＊ 組み込み機器を開発するメーカーが商用利用で組み込み Linux を利用する際には、組み込み Linux ベンダのものを利用するより、組み込みマザーボードに付属するものを利用するケースが多い。組み込みマザーボードメーカーが組み込み Linux ベンダから Linux を購入し、自社のマザーボードのデバイスドライバを書き、ツールチェーンを用意するからである。このように、組み込み機器メーカーより上流では、実際の現場でカーネル移植やデバイスドライバの開発をすることは稀になっている。

＊ 組み込み Linux システム開発を巡る環境は、アップトゥデートが激しいために、開発の情報はすぐに陳腐化する。最新の情報源については LinuxDevices.com にあたること。

3) 組み込み Linux の実装

＊ 組み込み Linux を OS として利用して、アプリケーションを開発する場合次の手順を経る。

- 1. Linux カーネル、デバイスドライバ、ブートローダーの選択
- 2. 開発環境の整備:
  GNU クロス開発ツールチェーン、カーネルソースコード、ブートコード等の入手
- 3. カーネルのターゲット移植
- 4. デバイスドライバの開発
- 5. ユーザランドの整備
- 6. アプリケーション開発
- 7. ROM/Flash 化、ファーム化
- 8. テスト

＊ ベンダの組み込みLinuxシステムを利用する場合は、1から5番までの作業が必要なくなるので、大幅な工数の削減が実現できる。とくに 3.カーネルをターゲット移植と 4.デバイスドライバの開発は難易度の高い作業である。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25. 組み込み開発環境に関する知識	応用
習得ポイント	II-25-10. 組み込み開発環境におけるデバッグの実際	
対応する コースウェア	第 15 回（組み込み開発環境におけるデバッグのパターン演習）	

II-25-10. 組み込み開発環境におけるデバッグの実際

組み込み開発環境において、プログラムをデバッグする手順を解説する。割り込み処理のデバッグ方法や、メモリ保護エラー、タイマーの不具合、通信データ異常など、様々なエラーに関する対処方法を説明する。

【学習の要点】

* 組み込みシステム開発においても、通常のソフトウェア開発同様に、デバグの基本はシンボリックデバッガを利用したステップ実行である。
* 割り込み処理、メモリ保護エラーのデバッグには、ハードウェアをチェックすることや、特定の命令の不正使用がないかチェックすることが必要な場合もある。
* タイマ不具合にはロジックアナライザの利用が、そして通信データの異常にはプロトコルアナライザの利用が必要な場合がある。
* ハードウェアデバッガを利用すると、CPU の命令実行を中止することなく、リアルタイムにレジスタやメモリの中身をトレースすることができる。

デバッグの実際

不具合	問題	対処
割り込み処理	特定の割り込み	ステップ実行
	全割り込み	全割り込み禁止命令の有無
	割り込みできない	ハードウェアチェック、割り込みクリア命令のチェック
メモリ保護エラー	特定の条件	ステップ実行
	長時間の運用	リソースモニタリング
	特定のハード	ハードの比較
タイマ不具合		タイマ割り込みチェック、ロジックアナライザの利用
通信データ異常		ステップ実行、プロトコルアナライザの利用

ハードウェアデバッガ（JTAGデバッガ）

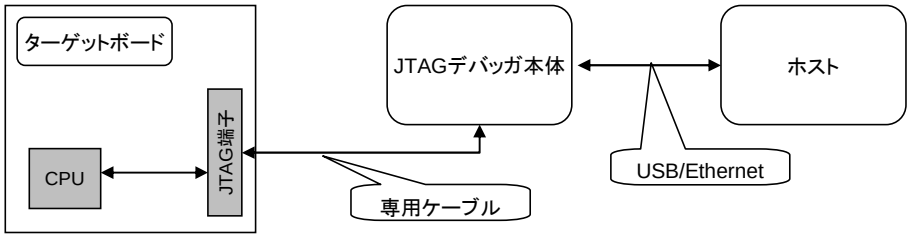


図 II-25-10.　デバッグの実際とハードウェアデバッガ

【解説】

1) デバッグの実際

＊ 割り込み処理の不具合

- 特定割り込み処理ができない場合は、該当箇所に、gdb 等のシンボリックデバッガを用いて、ブレークポイントを仕込み、ステップ実行を行う。
- 全く割り込みができない場合は、CPU 命令の全割り込み禁止命令がコードに書かれていないかチェックする。
- 割り込みばかりが生じてしまう場合は、接続しているハードウェアに問題がないか、また割り込みクリア処理を書き忘れていないかどうかをチェックする。

＊ メモリ保護エラー

- ある特定条件で発生する場合は、gdb 等のシンボリックデバッガを用いて、ブレークポイントを仕込み、ステップ実行を行い、メモリと変数をトレースする。
- 長時間の運用で発生する場合は、メモリリークなどの可能性が高いので、リソースモニタリングを行い、リソースの変化状況を捉え、その特徴をみてリーク箇所を推定する。
- 特定ハードウェアで起こる場合は、ハードウェアの比較試験を行う。

＊ タイマ不具合

- タイマ割り込みでタイマの異常が起きている場合は、他の割り込み処理に不具合が発生していないかチェックする。
- ロジックアナライザを利用して、タイマデバイスが故障していないか、また要求する精度に十分に達しているかを確認する。

＊ 通信データ異常

- 基本的なデバッギング技術はステップ実行であるが、原因の特定ができない場合は、プロトコルアナライザ等の機器を利用して、通信の処理をモニタリングする。

2) ハードウェアデバッガ

ハードウェアデバッガはハードウェアデバイスを利用してデバッグを行うツールである。

＊ ICE(In-Circuit Emulator)

- 実際の CPU 動作をエミュレートすることにより、CPU の動作を詳しく探る装置である。ホストから CPU の動作を自由に制御し、実行状況を観察することができる。CPU の命令実行を中止することなく、ブレークポイントを仕掛けることでリアルタイムにレジスタやメモリの中身をトレースすることができる。
- ICE を利用するには CPU をはずして、ICE プローブを装着する必要がある。
- ICE 本体は CPU アーキテクチャ依存で、異なった CPU には異なった ICE 本体が必要である。

＊ JTAG(Joint Test Action Group)デバッガ

- JTAG デバッガはターゲットボードにある JTAG 端子を経由して CPU と通信する。ホストからは JTAG デバッガを経由して、プログラムの任意の場所にブレークポイントを仕掛け、CPU のレジスタやメモリの内容をトレースすることができる。
- ICE よりも安価で、ボード上の JTAG 端子を利用するために、CPU を取り外す必要もない。