デバイスドライバの生成支援に関する考察

5Q－9

片山徹郎　山下勝也　最所圭三　福田晃
奈良先端科学技術大学院大学 情報科学研究科

1. はじめに

マルチメディアやインターネットの発展に伴い、多種多様なデバイスの登場が予測される。新しいデバイスに対応するには、オペレーティングシステム (OS) 内の既存のデバイスドライバを修正するか、デバイスドライバを新しく一から開発する必要がある。デバイスドライバの作成は、デバイスのハードウェアや、開発の対象となる OS に関する知識に加えて、タイミング制御などの複雑で注意深いコーディングを必要とするため、多大な労力を要する[1][2][3]。

そこで本稿では、デバイスドライバを作成する際にかかる負担を軽減するために、デバイスドライバの生成支援に関する考察を行なう。また、デバイスドライバ生成システムを提案し、そのシステムの入力形式について述べる。

2. デバイスドライバ生成システムの概要

デバイスドライバの作成には、多くの時間と労力が費される。デバイス、もしくは OS のいずれかが、変更されるたびに、それぞれに対応したデバイスドライバを作成しなければならない。さらに、同じサービスを提供するデバイスであっても、使用されているチップ (コントローラ) が異なれば、それに合った新たなデバイスドライバを作成しなければならない。

そこで、デバイスドライバを作成する際に、デバイスの種類、デバイスのハードウェアに依存した部所、OS に依存した部所を、それぞれ独立させる。具体的には、以下のものを用意する。

- デバイスドライバの仕様 — 生成するデバイスドライバが用意しなければならない関数、使用するデバイスを記述。
- デバイス依存仕様 — デバイスへの入出力動作やタイミング制御など、デバイスのハードウェアに依存する仕様。
- OS 依存仕様 — OS のデバイスドライバインターフェイスの定義。

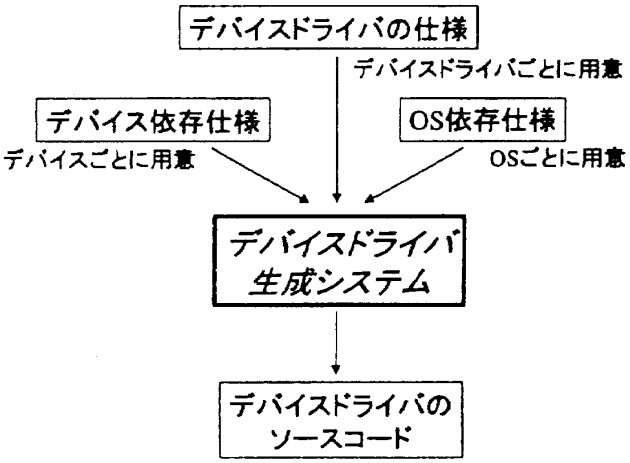

図 1: デバイスドライバ生成システムの概要

上記 3 つの仕様に基づいて、図 1のようなデバイスドライバ生成システムを提案する。デバイスドライバの仕様とデバイス依存仕様、OS 依存仕様とをデバイスドライバ生成システムに入力することにより、それぞれの仕様を満たすソースコードを生成する。

デバイスドライバの仕様とデバイス依存仕様と切り離すことにより、デバイス依存仕様に関しては再利用が可能である。逆に、新規にデバイスが開発された場合には、そのデバイス依存仕様のみを記述することにより、デバイスドライバを生成できる。

このことにより、あらかじめデバイスドライバの仕様が用意されているデバイスに対してデバイスドライバを作成する場合や、異なるアーキテクチャに移植する場合にかかる時間と手間を、大幅に削減することが期待できる。

3. システムの入力形式

デバイスドライバ生成システムには、デバイス依存仕様とデバイスドライバ仕様、OS 依存仕様の 3 つの入力が必要である。以下では、これらの入力形式について詳しく述べる。

- デバイスドライバの仕様 — デバイスドライバの構造を記述する。このため、以下の内容を記述する必要がある。

  – デバイスドライバの名前の宣言

Support to Generate Device Drivers.
Tetsuro Katayama, Katsuya Yamashita, Keizo Saisho and Akira Fukuda
Graduate School of Information Science,
Nara Institute of Science and Technology.

```
driver_name "ep"

%<stop>{
if (sc->gone) {
    return;
}

%<ep.stop>
}
```

(a) デバイスドライバの仕様

```
out16 /{ BASE + EP_COMMAND /} /{ RX_DISABLE /};
out16 /{ BASE + EP_COMMAND /} /{ RX_DISCARD_TOP_PACK /};
while (in16 /{ (BASE + EP_STATUS) & S_COMMAND_IN_PROGRESS /} );
out16 /{ BASE + EP_COMMAND /} /{ TX_DISABLE /}
out16 /{ BASE + EP_COMMAND /} /{ STOP_TRANSCEIVER /};
out16 /{ BASE + EP_COMMAND /} /{ RX_RESET /};
out16 /{ BASE + EP_COMMAND /} /{ TX_RESET /};
while (in16 /{ (BASE + EP_STATUS) & S_COMMAND_IN_PROGRESS /} );
out16 /{ BASE + EP_COMMAND /} /{ C_INTR_LATCH /};
out16 /{ BASE + EP_COMMAND /} /{ SET_RD_0_MASK /};
out16 /{ BASE + EP_COMMAND /} /{ SET_INTR_MASK /};
out16 /{ BASE + EP_COMMAND /} /{ SET_RX_FILTER /};
```

(b) デバイス依存仕様

```
static void %<driver.name>_stop __P((struct %<driver.name>_softc *sc))
```

(c) OS 依存仕様

図 2: stop 関数における仕様の記述例

  - OS 側が要求するデバイスドライバインターフェイスの宣言
  - デバイス依存仕様とのインタフェースの宣言
  - デバイスドライバ内の関数のテンプレート
  - デバイスドライバ内で使用するローカル変数および構造体の宣言と定義

- デバイス依存仕様 — デバイスに依存した部分を記述する。このため、以下の内容を記述する必要がある。
  - デバイスとのデータの入出力動作
  - タイミング制御
  - ハードウェア割り込み
  - OS とデータを受渡しするための機構

- OS 依存仕様 — OS に依存した部分を記述する。このため、以下の内容を記述する必要がある。
  - OS 側が要求するデバイスドライバインターフェイスの定義
  - デバイスドライバに必要なヘッダファイルおよびデータ構造の定義
  - ハードウェアを直接制御する関数の定義

図 2 は、OS を FreeBSD[4]、デバイスを 3Com 社のネットワークデバイス Etherlink III、デバイスドライバの stop 関数について、それぞれの仕様を記述したものである。仕様を記述する言語には我々が定義した中間言語を用いている[3]。

4. おわりに

本稿では、デバイスドライバの生成支援に関する考察を行なった。デバイスドライバ生成システムを提案し、システムの入力形式について述べた。システムは、デバイスドライバの仕様、デバイス依存仕様、OS 依存仕様の 3 つを入力とし、その 3 つを満たすデバイスドライバのソースコードを出力する。

デバイス依存仕様と OS 依存仕様は再利用が可能である。すなわち、あらかじめ用意されたデバイス依存仕様を利用して、あるデバイスから別の OS に対するデバイスドライバを作成することができる。逆に、OS 依存仕様を用いて、同じ OS 上の異なるデバイスのデバイスドライバを作成することができる。また、デバイス開発者がデバイス依存仕様を用意し、OS 開発者が OS 依存仕様を用意すれば、デバイスドライバの作成作業が分担できる。以上のことから、デバイスドライバを作成する際にかかる時間および手間の削減が期待できる。

今後は、デバイスドライバ生成システムを多くのデバイスおよび OS に対応させ、本手法の有効性を確認する。

参考文献

[1] P.H. Chou, R.B. Ortega, G. Borriello: "The Chinook Hardware/Software Co-synthesis System," *Proc. 8th Int. Sympo. on System Synthesis*, pp.22-27, 1995.

[2] 片山徹郎, 最所圭三, 福田晃: "デバイスドライバの自動生成に向けて –プリンタデバイスの生成に関する考察–," 情報処理学会研究報告, 97-OS-76, pp.43-48, 1997.

[3] 山下勝也, 片山徹郎, 最所圭三, 福田晃: "デバイスドライバ生成システムにおける入力形式に関する考察," 情報処理学会研究報告, 98-OS-79, 1998 (発表予定).

[4] FreeBSD Inc: http://www.freebsd.org/