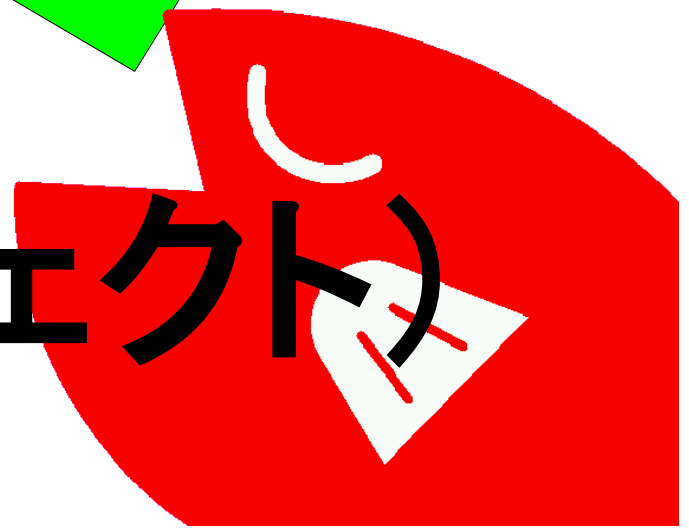
リンカとローダで遊んでみよう！

～知られざるリンクの世界～

坂井弘亮
(KOZOSプロジェクト)



資料は以下にあります

http://www.saturn.dti.ne.jp/~hsakai/linker.html

聞きながら試したい人は
サンプルコードを
ダウンロードしてください
(FreeBSD用です)

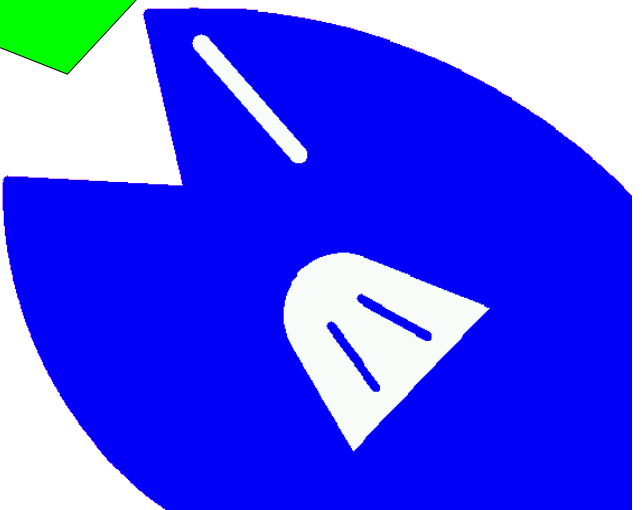
略歴

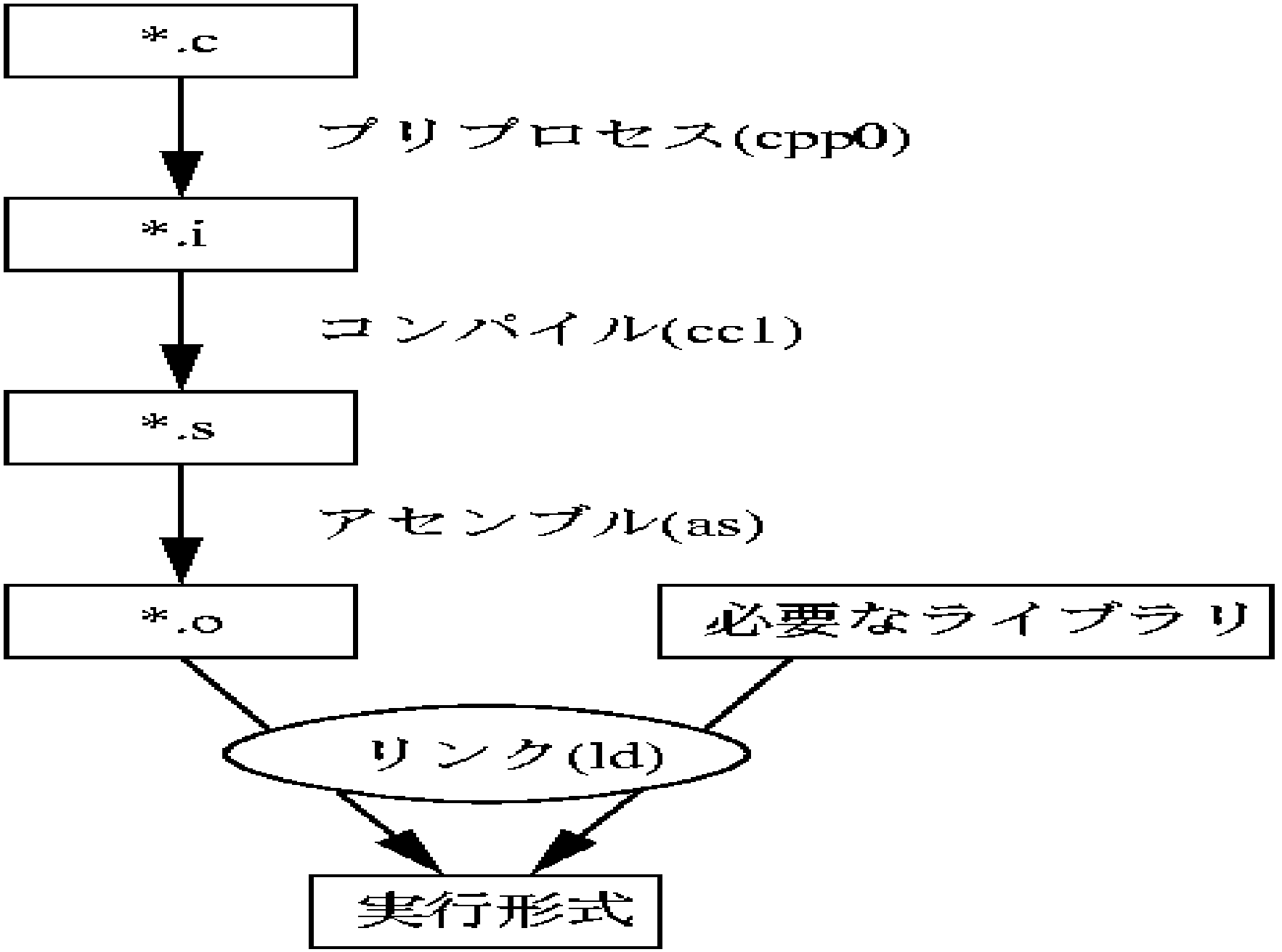
1997年頃 プログラミングに傾倒
X Window System プログラミング
オブジェクト指向プログラミング
OSに興味を持つ
1999年 就職
組込みシステムに興味を持つ
2007年 自作組込みOS「KOZOS」の開発を
趣味で開始
2009年 OSC出展
KOZOSを実ボードに移植
2010年「12ステップで作る 組込みOS自作入門」
出版

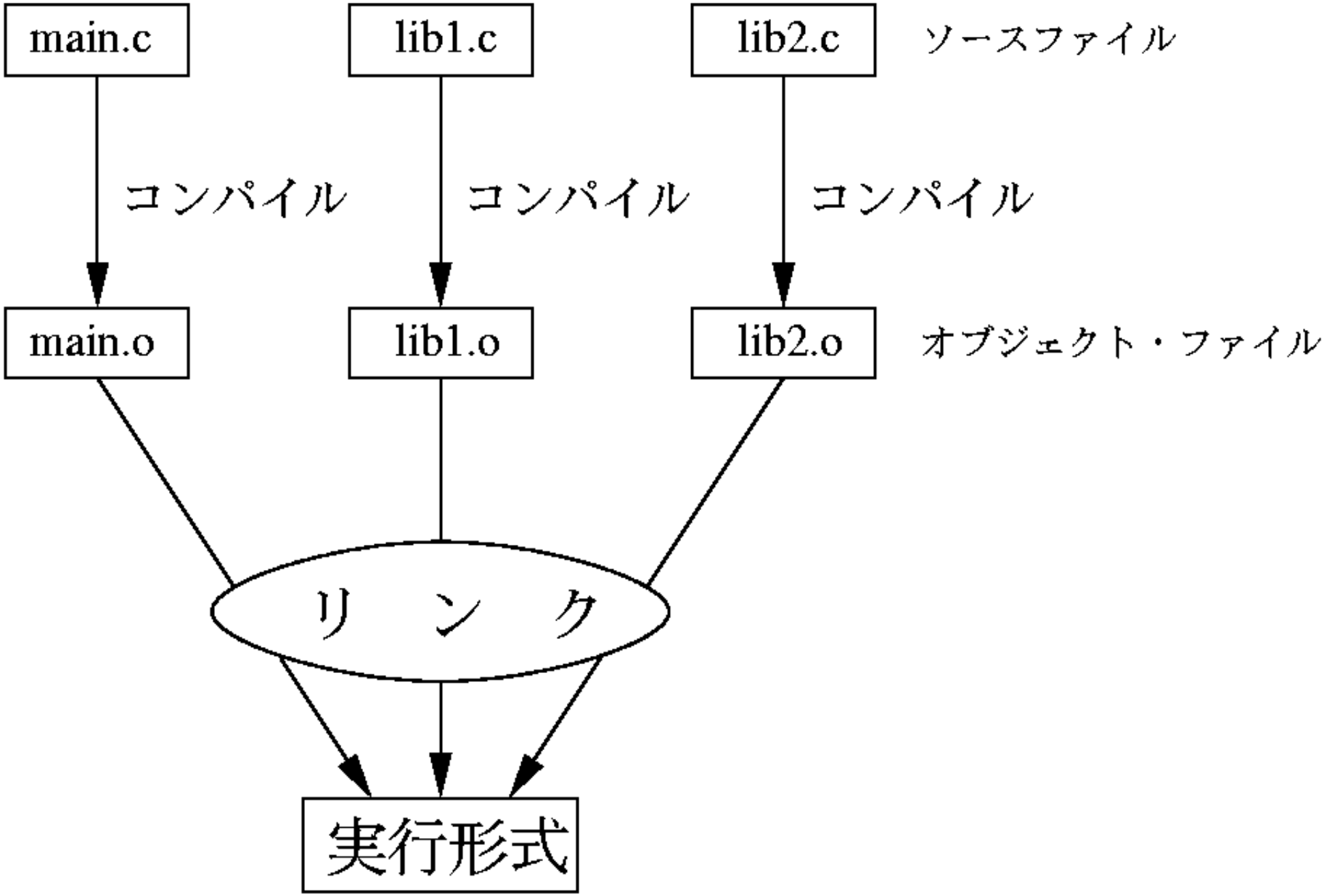
基礎知識の説明後、
デモをします
リンカやローダで
いろいろ試したり
遊んでみたりします

1. テキスト、データ、BSS領域
2. オブジェクトファイルフォーマット、
   ELFフォーマット、セクション
3. リンカスクリプト
4. コア・ファイル
5. シンボル、再配置

そもそも
「リンカ」って何？



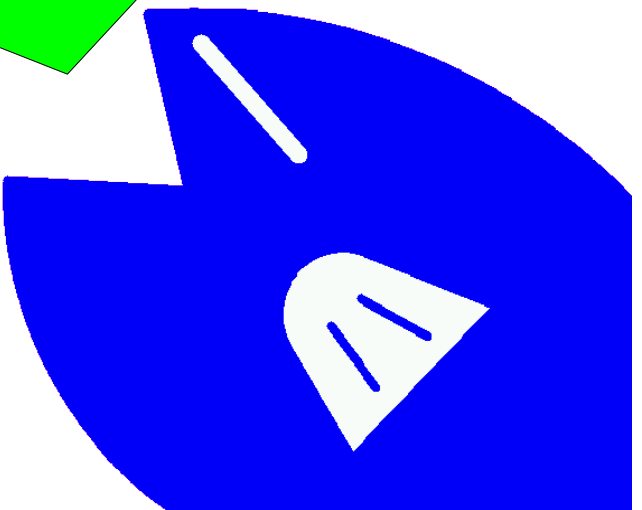

*.c
プリプロセス(cpp0)
*.i
コンパイル(cc1)
*.s
アセンブル(as)
*.o
必要なライブラリ
リンク(ld)
実行形式


main.c
lib1.c
lib2.c
ソースファイル
コンパイル
コンパイル
コンパイル
main.o
lib1.o
lib2.o
オブジェクト・ファイル
リ　ン　ク
実行形式

実行形式ファイルを作成するために…

- 複数のオブジェクトファイルをまとめる
- ライブラリもまとめる
- ファイルをまたいだ関数呼び出しを解決する

じゃあ、
「ローダ」って何？



- CPUが実行できるのは、メモリ上の機械語コードのみ
- プログラムを実行するには、HDD等からメモリ上に展開する必要がある（ロード）
- ロードを行うためのプログラムを「ローダ」という
- いわゆる「ブートローダー」もローダの一種

なぜ今「リンカ」が熱いのか…？

- OSのカーネルのソースなどでは、リンカやローダの機能を駆使した書き方をされることもある

- 組込みではメモリマップを意識したプログラミングが必要になるので、リンカやローダの知識が要る

- ソフトウェア・セキュリティの分野では
  実行形式をバイナリハックして
  脆弱性を解析したりすることも

- 最近はJITコンパイラや動的最適化
  のために必要になることも…？

- バイナリハックとか、流行りだし
  （バイナリアンのかた、いかがで
  しょうか？）

- まあでも、このへんのことも
  知らないとなんか気になるし
  面白そうだし知りたいじゃん!

- リンカとローダを駆使すると、
  C言語のレベルでは不可能
  なおもしろいことができる！

(注)実用的かどうかは別問題

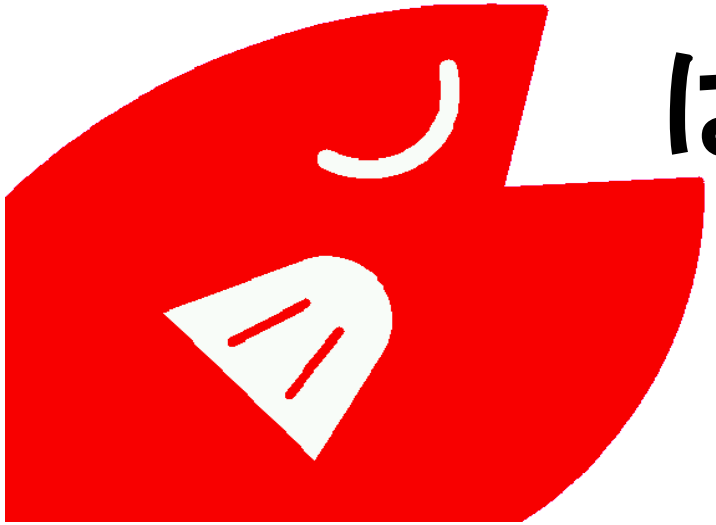
1
テキスト、データ、BSS領域

実行形式には3つの領域があります

- テキスト領域
  …機械語コードが配置される
- データ領域
  …静的変数が配置される
  ロードのときに展開される
- BSS領域
  …初期値無しの静的変数が
  配置される
  ロードのときにゼロクリアされる

```
int value = 1; …データ領域へ
int value2;    …BSS領域へ

int main()     …テキスト領域へ
{
    …;
}
```



はいここで実験です！

- const変数書き換え
- 文字列リテラル書き換え

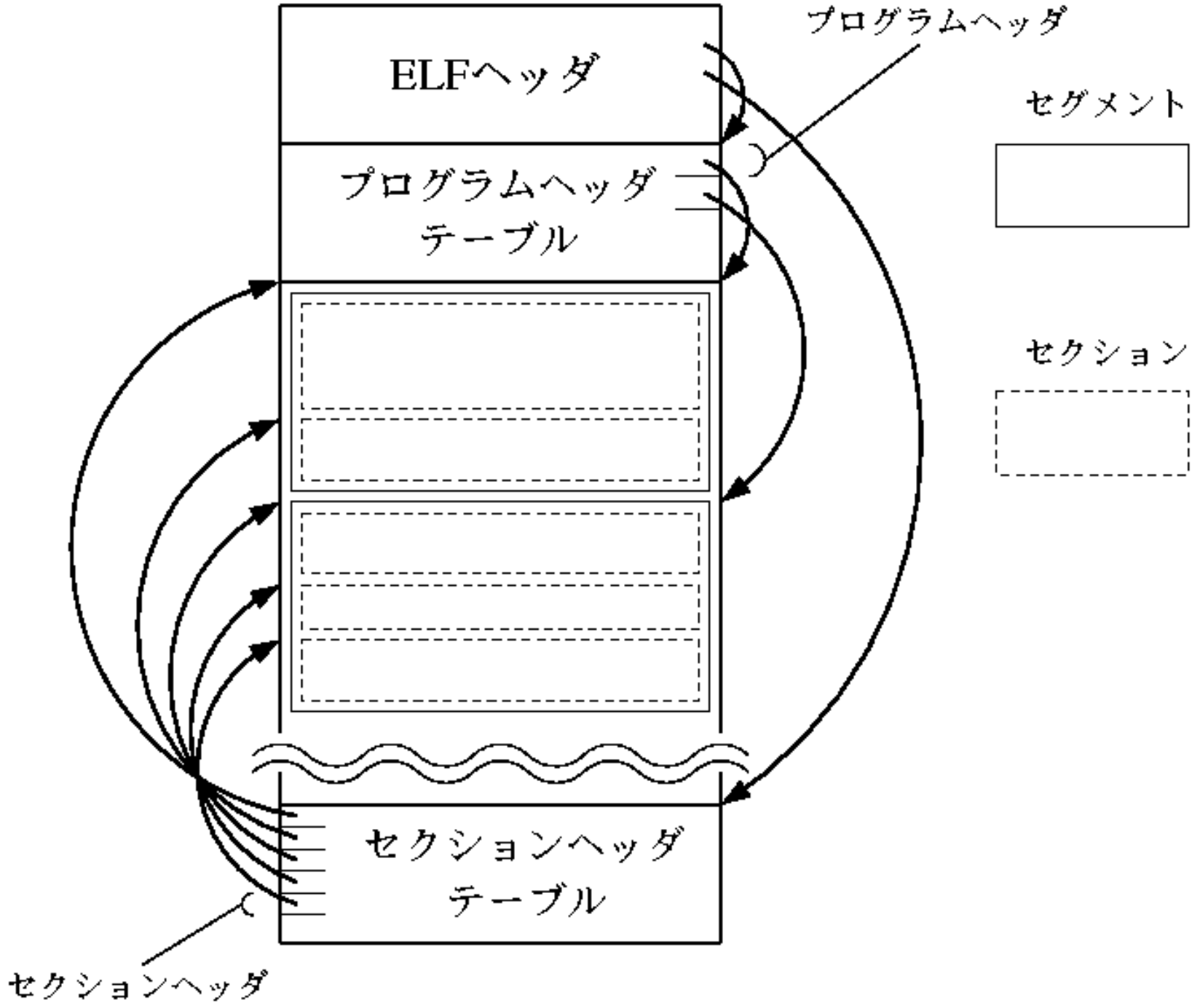
2.
オブジェクトファイルフォーマット、ELFフォーマット、セクション

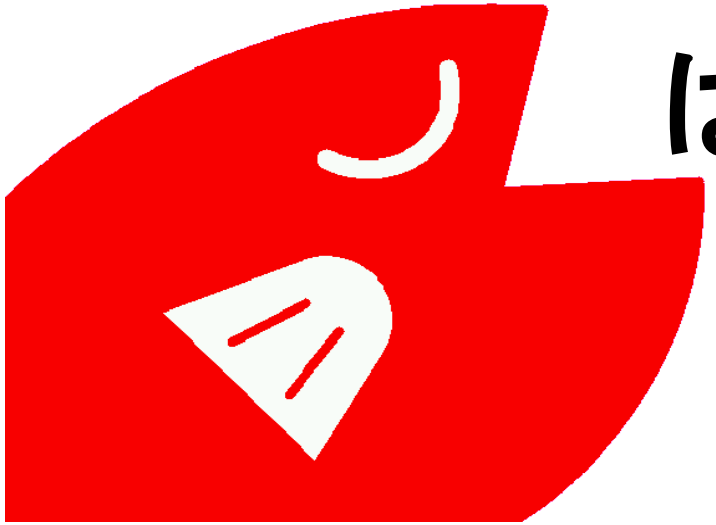
・オブジェクトファイルフォーマット
　…オブジェクトファイルや実行形式のファイルフォーマット

・a.out形式、ELF形式、COFF形式など

ELF形式

- オブジェクトファイル、実行形式、コアファイル、共有ライブラリが表現可能
- バイエンディアン、32/64ビット対応
- セクションとセグメントの細かい設定
- かなり万能のフォーマット


ELFヘッダ
プログラムヘッダ
テーブル
セクションヘッダ
テーブル
プログラムヘッダ
セグメント
セクション
セクションヘッダ



はいここで実験です！

- ELF形式の解析
- セクションの作成
- バイナリデータ埋め込み
- 簡易ローダ作成

3.
リンカスクリプト

- リンカスクリプト
  …セクションの配置などを
  細かくチューニングできます

- 組込みとかでは必須なのに
  あまり資料が無い！

リンカスクリプトを見てみよう！

リンカスクリプトのファイル名は，様々．
拡張子が決まっていないので，いろんな名前になっている．

例(ちょっと古い情報です)

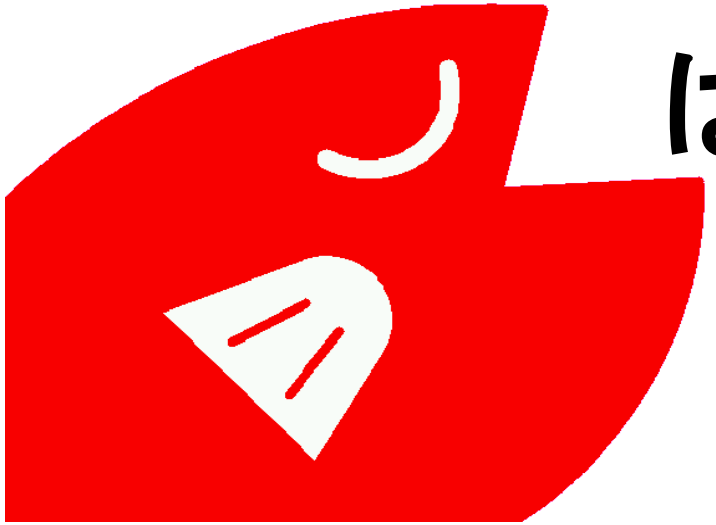
FreeBSDのカーネル
/usr/src/sys/conf/ldscript.i386, ldscript.alpha

FreeBSD のアプリケーション用
/usr/libdata/ldscripts/elf_i386.x, elf_i386.xn, elf_i386.xs

Linux のカーネル
arch/$(ARCH)/kernel/vmlinux.lds.S
(この他にもカーネルローダ用やramdisk用などが点在している)

Linux のカーネルローダ
arch/i386/boot/compressed/vmlinux.scr
arch/sh/boot/compressed/vmlinux.scr
arch/arm/boot/compressed/vmlinux.lds.in
arch/ppc/boot/ld.script

NetBSD のカーネル
sys/arch/mmeye/conf/sh.x
sys/arch/i386/conf/kern.ldscript
sys/arch/sparc/conf/kern.ldscript
sys/arch/mips/conf/kern.ldscript
(この他にもカーネルローダ用が点在しているようだ)



はいここで実験です！

- バイナリデータ埋込み(2)
- 静的変数の再初期化

ここでちょっと
紹介タ～イム！

組込み勉強会
やります！

「組込みこぞう勉強会」

９月２５日(土)
東京近郊
良い場所、探してます

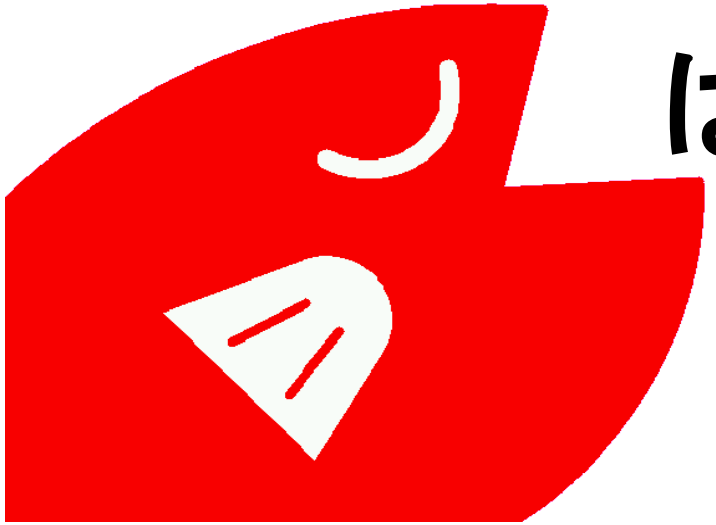
宣伝終わり！
本編に戻ります

4.
コア・ファイル

コア・ファイルとは？
アプリが不正メモリアクセス
とかでダウンしたときに、
生成されるファイル
「*.core」ってやつ
(「コア」という名前が
歴史を感じさせる)

コア・ファイルもELF形式だって知ってた？

ダウンしたときの
メモリのダンプとかが
格納されています



はいここで実験です！

- coreファイル解析
- coreファイルからの動作再開

5.
シンボル、再配置

再配置

コンパイラが作成した
オブジェクトファイルでは、
ファイル間をまたいだ関数の
呼び出しや変数の参照部分は、
空欄になっています
(アドレスが決定されていないため)

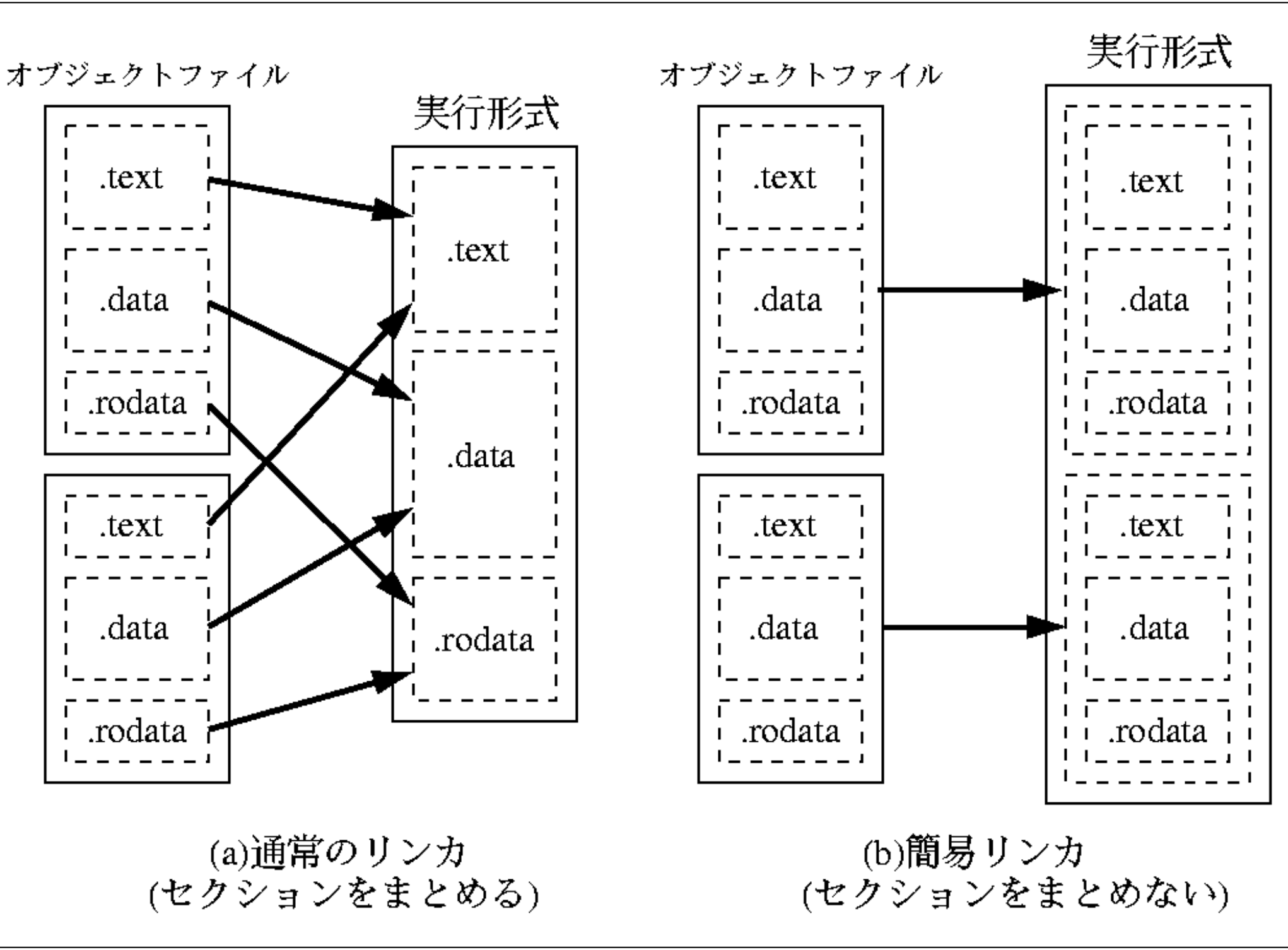
→リンク時に決定されます(再配置)
→リンク時に補填されます(名前解決)

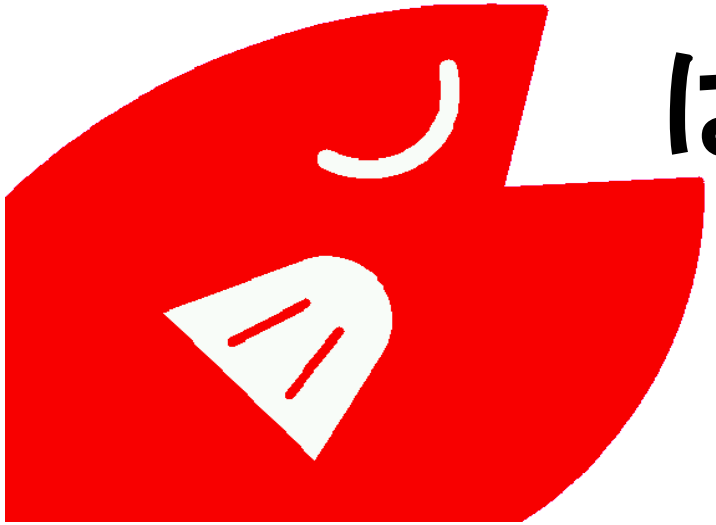
再配置のためには…

変数の参照や
関数の呼び出しごとに、
どの変数や関数を
呼び出すのかの
情報が必要
（再配置エントリ）

```
void func()
{
  extern int value;
  value = 1;      …valueの再配置エントリが必要
  func2();        …func2の再配置エントリが必要
}

int main()
{
  func();         …funcの再配置エントリが必要
}
```

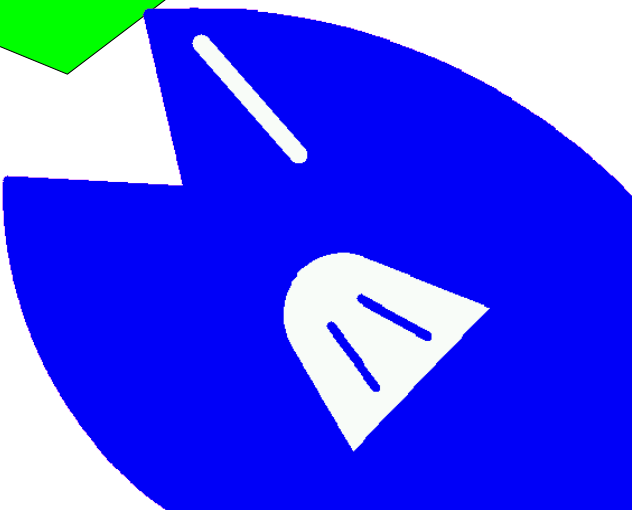

オブジェクトファイル
.text
.data
.rodata
.text
.data
.rodata
実行形式
.text
.data
.rodata
(a)通常のリンカ
(セクションをまとめる)
オブジェクトファイル
.text
.data
.rodata
.text
.data
.rodata
実行形式
.text
.data
.rodata
.text
.data
.rodata
(b)簡易リンカ
(セクションをまとめない)



はいここで実験です！

- 再配置の実際
- 簡易動的リンカ作成
- 簡易静的リンカ作成

最後にちょっと
役に立つものを
…



不要シンボルのチェックツール (checksym)

オブジェクトファイルを与えると…

利用されていない変数、関数の一覧を出力してくれます

ちょっと試してみましょう

そろそろ、
まとめに
入りますが

どうでしたか？
リン力って
面白いでしょ？

でも、リンカやローダの
情報って、あまり無い
（ネット上も、書籍も）

あったとしても、
単発だったり、
他の情報のおまけの
内容だったりして、
まとまっていない

リンカやローダの
専門的な情報として、
まとめられていない！

OSを
いじろうとしたときに、
これが一番困った

なんとか
ならんもんか…

じゃ、書くか!

CQ出版より8月9日発売!

各方面から
賛辞の声が！

「これは、素晴らしく
マニアックですねえ」

「相変わらず、
マニアックなものを
書きますねえ」

「これからもぜひ、
マニアックなものを
書き続けてください」

ほめられた！

みんな、
バイナリハックが
好きなんだなあ…

おしまい