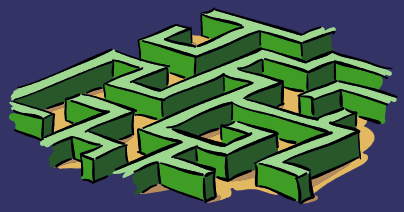
getcontextの怪

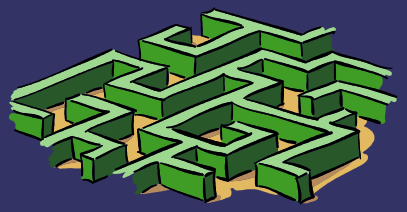
田中 哲
akr@fsij.org

Binary 10.0 カンファレンス
11111010101-1100-01111



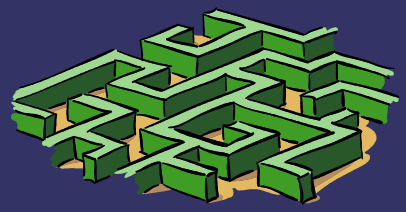
概要

- Ruby が IA-64 でうまく動かなかった
- getcontext問題ではgccを変えてもらった



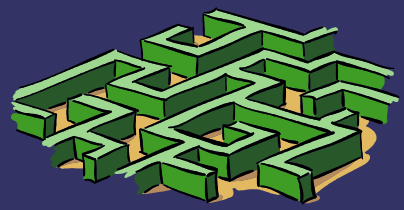
Ruby

- getcontext を使うこともあるプログラム
- 昔は IA-64 でうまく動かなかった
- 今は大きな問題はない



Ruby の *IA-64* 対応

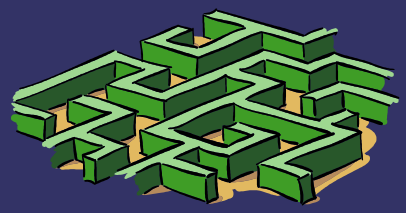
- 2003-07 対応開始
- 2003-07 [ruby-dev:20788] SEGV (GNU/Linux, gcc)
- 2004-01 [ruby-dev:22634] SEGV (GNU/Linux, gcc)
- 2004-03 [ruby-talk:95256] SEGV (2004-03-15, GNU/Linux, gcc)
- 2004-07 [ruby-talk:107469] SEGV (GNU/Linux, gcc)
- 2004-07 [ruby-talk:107613] SEGV / unexpected break (GNU/Linux, gcc)
- 2004-08 [ruby-talk:110682] SEGV (2004-08-26, GNU/Linux, gcc)
- 2005-03 [ruby-talk:142863] SEGV (1.8.2, GNU/Linux, gcc)
- 2005-05 [ruby-core:4830] unexpected break (2005-05-02, GNU/Linux, gcc)
- 2006-03 [ruby-core:7641] [BUG] Bus Error (1.8.4, HP-UX, gcc)
- 2006-04 [ruby-core:7687] [ruby-Bugs-4047] SEGV (1.8.4, HP-UX, HP C)



いろいろな症状

- IA-64 で getcontext を使うと問題が起きる
- getcontext じゃなくて setjmp なら問題が起きない
- SPARC 上の gcc でも問題が起こる
- Sun のコンパイラでは問題が起きない
- Intel Compiler でも問題が起きる
- HP のコンパイラでは問題が起きない
- FreeBSD/x86 でも問題が起きる

???????



getcontext の使用例

```
#include <ucontext.h>

int flag;
ucontext_t cont;

void f(void)
{
  flag = 1;
  setcontext(&cont);
}
```

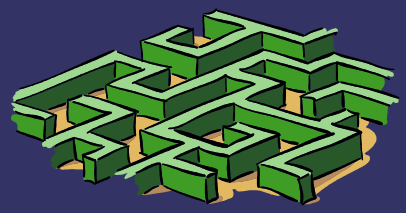
```
int g(void)
{
  int ret;
  flag = 0;
  getcontext(&cont);
  ret = flag;
  if (ret == 0) {
    f();
  }
  return ret;
}

int main(int argc, char **argv)
{
  g();
  return 0;
}
```

実行結果

```
% gcc -O2 -g tst.c
% ./a.out
zsh: segmentation fault  ./a.out
```

祝! SEGV!!

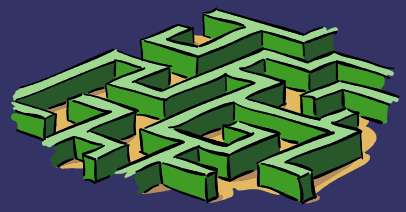
gcc 3.3.5 on Debian GNU/Linux sarge (ia64)



getcontext/setcontext

```
#include <ucontext.h>
int getcontext(ucontext_t *ucp);
int setcontext(const ucontext_t *ucp);
```

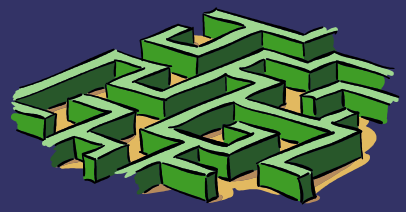
- setjmp/longjmp に類似した関数
- setjmp ≈ getcontext
- longjmp ≈ setcontext
- 返り値が違う (成功・失敗しか示さない)
- makecontext/swapcontext まで使うと面白いが今日の話には関係ない



setjmp/longjmp

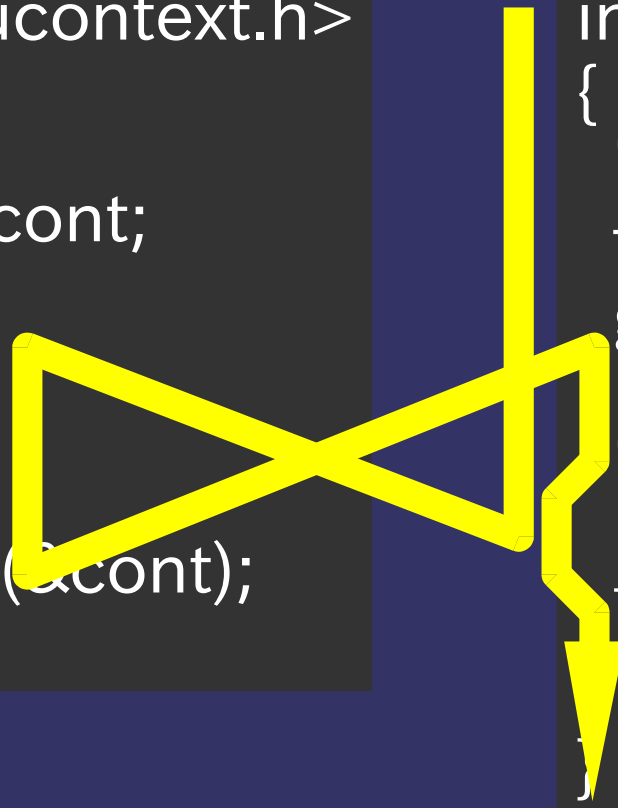
```
#include <setjmp.h>
int setjmp(jmp_buf env);
void longjmp(jmp_buf env, int val);
```

- setjmp: レジスタの内容を env に保存
- longjmp: env からレジスタの内容を復元
- longjmp から setjmp へ値を伝えられる (0以外)
- 大域脱出が可能
  - プログラムカウンタが復元する
  - スタックポインタも復元する
- メモリは復元しない



正常な動作の流れ

```
#include <ucontext.h>

int flag;
ucontext_t cont;

void f(void)
{
  flag = 1;
  setcontext(&cont);
}
```



```
int g(void)
{
  int ret;
  flag = 0;
  getcontext(&cont);
  ret = flag;
  if (ret == 0) {
    f();
  }
  return ret;
}

int main(int argc, char **argv)
{
  g();
  return 0;
}
```

SEGV時の流れ

```
#include <ucontext.h>

int flag;
ucontext_t cont;

void f(void)
{
  flag = 1;
  setcontext(&cont);
}
```

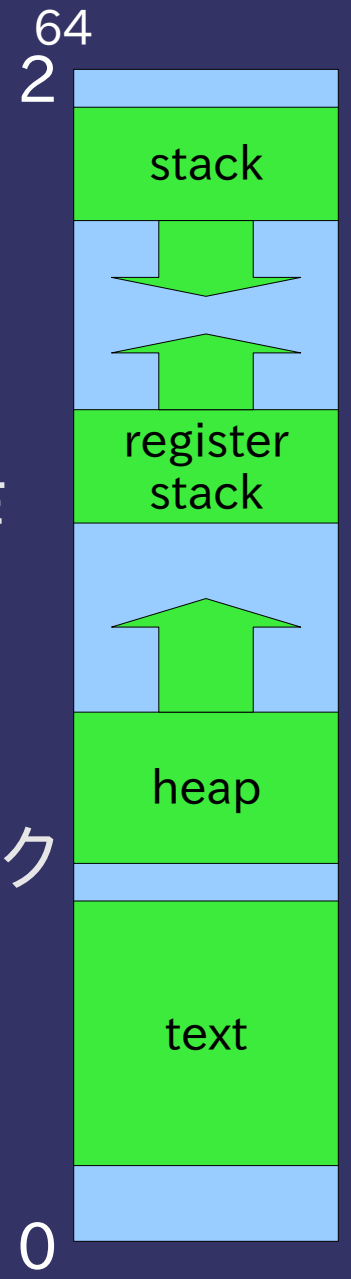
```
int g(void)
{
  int ret;
  flag = 0;
  getcontext(&cont);
  ret = flag;
  if (ret == 0) {
    f();
  }
  return ret;
}

int main(int argc, char **argv)
{
  g();
  return 0;
}
```

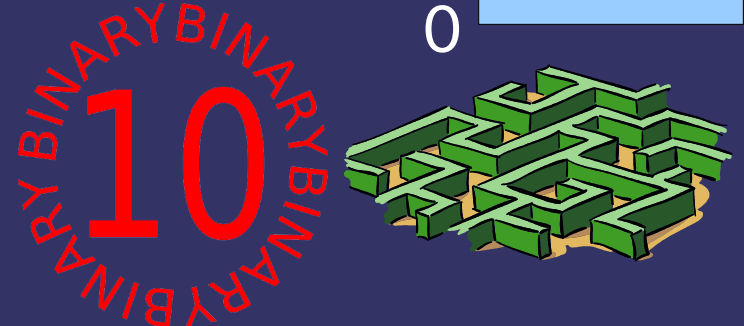
SEGV

ret = flag;
でSEGV?

IA-64 (Itanium)

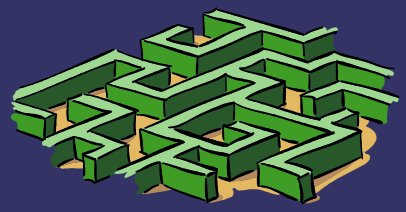
- Intel の 64bit プロセッサ
- HP-UX, GNU/Linux, FreeBSD などが動作
- たくさんレジスタがある
  - 汎用レジスタだけで 128個 (r0 ~ r127)
- レジスタスタック
  - 関数呼び出しでレジスタを保存する専用のスタック
  - r32 ~ r127 の 96個を自動的に退避・復帰する
  - レジスタの退避・復帰コードが不要
  - 退避・復帰はメモリが暇なときに適当にやる
  - SPARCのレジスタウインドウに類似





IA-64 での *setjmp/getcontext*

- setjmpはレジスタスタックのレジスタ(の中身)は保存･復元しない
  Itanium Software Conventions and Runtime Architecture Guide
- getcontextについては記載がない
  glibc の実装は保存･復元しない
- 理由?
  - 関数呼び出しで破壊されない
  - レジスタをたくさん保存するのは手間

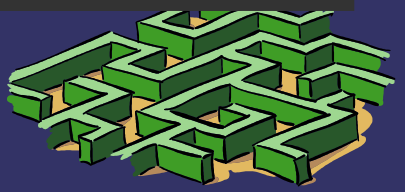


ret = flag で SEGV

```
flag = 0;
getcontext(&cont);
ret = flag;
```

コンパイル結果

```
ld8.mov r32 = [r14], flag#
st4 [r32] = r0
br.call.sptk.many b0 = getcontext#
ld4 r32 = [r32]
```



ret = flag で SEGV

```
flag = 0;
getcontext(&cont);
ret = flag;
```

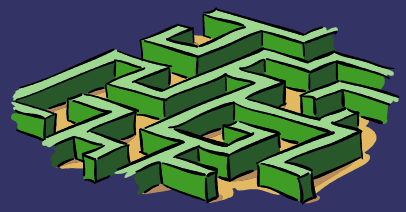
Cっぽく書くと

```
r32 = &flag;
*r32 = 0;
getcontext();
r32 = *r32;
```

r32が flag の
アドレスじゃなくなってる!

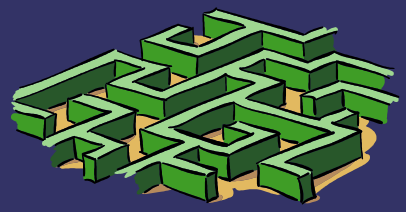
SEGV のしくみ

- getcontext 呼び出しの前に r32 を設定している
- getcontext から返ってきてから r32 を使っている
- その後で setcontext を呼び出すまでに r32 を変更している
- setcontext から getcontext 直後に飛んだとき、r32 の値は変更後の値になっている
- その値を変更前の値だと思って使うとマズイ



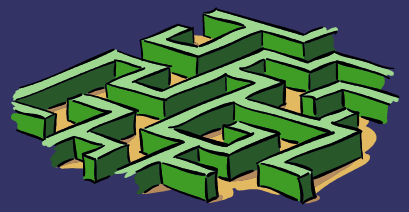
誰が悪い? どこに報告する?

- glibc の getcontext は r32 を保存しない
  - setjmp との一貫性か
  - glibc で直すなら r32 を保存してもらう?
  - レジスタスタック全体を保存する?
- gcc は関数呼び出し前後で r32 は不変と仮定
  - レジスタスタックはその仮定を実現するためにある
  - gcc で直すなら r32 を使わないようにしてもらう?
  - レジスタスタック全体を使えなくなる?
- アプリケーションで対処する?
  - 変数のアドレスの扱いは制御困難



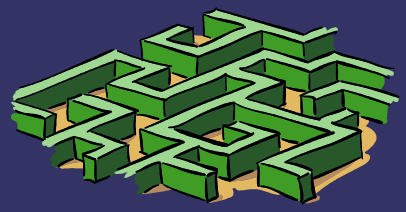
報告と反応 *(IA-64)*

- 2005-06-06: Debian Bug#312188
  glibc の問題として報告 → 反応無し
- 2005-06-08: GCC Bugzilla Bug 21957
  IA-64 での問題として報告 → 反応無し



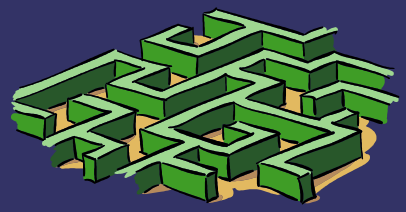
報告と反応 *(SPARC)*

- 2005-06-20: GCC Bugzilla Bug 22127 SPARC での問題として報告 → 反応有り
  2005-11 setjmp 用の細工が getcontext にも有効になった
  2006-02-28 gcc 4.1.0
  2006-03-10 gcc 4.0.3



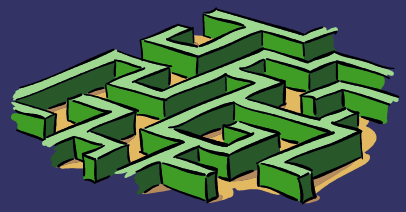
報告と反応 (他)

- 2005-12-30: Intel Software Network forum で icc の問題として報告 → 反応無し
- 2006-01: FreeBSD PR#92110
  - コードを整理でなぜか FreeBSD/x86 で問題発生 → 直してもらう



いろいろな症状

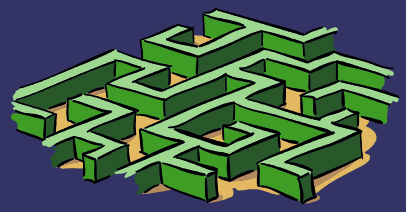
- IA-64 で getcontext を使うと問題が起きる
- getcontext じゃなくて setjmp なら問題が起きない
- SPARC 上の gcc でも問題が起こる
- Sun のコンパイラでは問題が起きない
- Intel Compiler でも問題が起きる
- HP のコンパイラでは問題が起きない
- FreeBSD/x86 でも問題が起きる



まとめ

- ときには下にいかないとわからないこともある
- レジスタスタックなんてなければいいのに
- 誰が悪いのかよくわからない
- でも gcc に対処してもらった

- setjmp/getcontext はコンパイラが知っていなければならない
  - 大域脱出をライブラリにしたのは間違っていた
- getcontext みたいな一般的でないのを使うから問題になる



BINARY BINARY BINARY BINARY

10