日曜研究室

技術的な観点から日常を綴ります

[xv6 #32] Chapter 2 – Traps, interrupts, and drivers – Exercises

テキストの42ページ

本文

1. 最初のシステムコールをキャッチするために、syscall()の最初の命令にブレークポイントをセットしなさい。
（例えば、GDBのbr syscallコマンドを使って）
この時点のスタック上の値は何か？
このブレークポイントにおけるx/37x $espの出力を、何のためのものかラベル付けして説明しなさい。
（例えば、%ebpはtrap, trapframe.eip, 作業用スペース等）

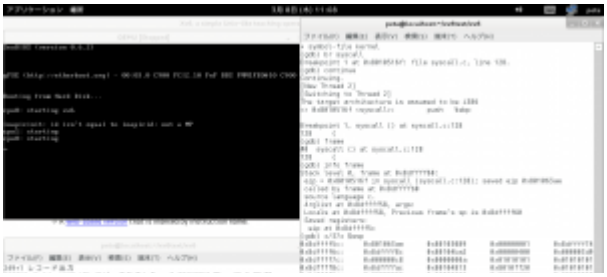
2. 新たにシステムコールを追加しなさい。

3. ネットワークドライバを追加しなさい。

作業

1. について
コンパイル～GDBによる追跡の方法についてはその0とその20を参照。

GDBでx/37x $espをやった時点のスクリーンショットはこんな感じです。



GDBの出力の重要な部分をテキストで抜き出すと、

```
(gdb) info frame
Stack level 0, frame at 0x8dffff60:
 eip = 0x8010516f in syscall (syscall.c:128); saved eip 0x801063ae
 called by frame at 0x8dffffb0
 source language c.
 Arglist at 0x8dffff58, args:
 Locals at 0x8dffff58, Previous frame's sp is 0x8dffff60
 Saved registers:
  eip at 0x8dffff5c
(gdb) x/37x $esp
0x8dffff5c: 0x801063ae  0x80103089  0x00000001  0x8dffff74
0x8dffff6c: 0x8dffff8c  0x80104ba2  0x00000400  0x000003d9
0x8dffff7c: 0x000000c8  0x0000000a  0x01010101  0x01010101
0x8dffff8c: 0x8dffffac  0x80104813  0x8010ff20  0x01010101
0x8dffff9c: 0x00000000  0x00000000  0x00000000  0x00000000
0x8dffffac: 0x80106199  0x8dffffb4  0x00000000  0x00000000
0x8dffffbc: 0x00000000  0x8dffffd4  0x00000000  0x00000000
0x8dffffcc: 0x00000000  0x00000007  0x00000000  0x00000000
0x8dffffdc: 0x0000002b  0x0000002b  0x00000040  0x00000000
0x8dffffec: 0x00000013
(gdb)
```

正直さっぱりですが、その25に載せた図2–2に当てはめるといいのかな。
その27から状況証拠（笑）的には0x00000007はSYS_execを表してるんだろうなと思います。

2. について
簡単にするためにシステムに依存しないシステムコールとして階乗を計算するシステムコールを追加してみました。
無意味ですね〜。
それと10の階乗を求めるテストプログラムも追加してます。

xv6-rev6とのdiffはこんな感じ。

```
diff --git a/Makefile b/Makefile
index ffb085f..2b93723 100644
--- a/Makefile
+++ b/Makefile
@@ -27,6 +27,7 @@ OBJS = \
    uart.o\
    vectors.o\
    vm.o\
+   factorial.o\

 # Cross-compiling (e.g., on Mac OS X)
 #TOOLPREFIX = i386-jos-elf-
@@ -72,8 +73,8 @@ AS = $(TOOLPREFIX)gas
 LD = $(TOOLPREFIX)ld
 OBJCOPY = $(TOOLPREFIX)objcopy
 OBJDUMP = $(TOOLPREFIX)objdump
-#CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -O2
```

```
    -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer
018 -CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -Wall
    -MD -ggdb -m32 -Werror -fno-omit-frame-pointer
019 +#CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -O2
    -Wall -MD -ggdb -m32 -fno-omit-frame-pointer
020 +CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -Wall
    -MD -ggdb -m32 -fno-omit-frame-pointer
021  CFLAGS += $(shell $(CC) -fno-stack-protector -E -x c /dev/null
    >/dev/null 2>&1 && echo -fno-stack-protector)
022  ASFLAGS = -m32 -gdwarf-2 -Wa,-divide
023  # FreeBSD ld wants ``elf_i386_fbsd''
024 @@ -146,7 +147,7 @@ _forktest: forktest.o $(ULIB)
025      $(OBJDUMP) -S _forktest > forktest.asm
026
027  mkfs: mkfs.c fs.h
028 -    gcc -m32 -Werror -Wall -o mkfs mkfs.c
029 +    gcc -m32 -Wall -o mkfs mkfs.c
030
031  UPROGS=\
032      _cat\
033 @@ -164,6 +165,7 @@ UPROGS=\
034      _usertests\
035      _wc\
036      _zombie\
037 +    _factorialtest\
038
039  fs.img: mkfs README $(UPROGS)
040      ./mkfs fs.img README $(UPROGS)
041 diff --git a/defs.h b/defs.h
042 index 921c7bf..02cf6be 100644
043 --- a/defs.h
044 +++ b/defs.h
045 @@ -178,5 +178,8 @@ void            switchkvm(void);
046  int             copyout(pde_t*, uint, void*, uint);
047  void            clearpteu(pde_t *pgdir, char *uva);
048
049 +// factorial.c
050 +int        factorial(int);
051 +
052  // number of elements in fixed-size array
053  #define NELEM(x) (sizeof(x)/sizeof((x)[0]))
054 diff --git a/factorial.c b/factorial.c
055 new file mode 100644
056 index 0000000..ae4e1cb
057 --- /dev/null
058 +++ b/factorial.c
059 @@ -0,0 +1,25 @@
060 +#include "types.h"
061 +#include "defs.h"
062 +#include "param.h"
063 +#include "stat.h"
064 +#include "mmu.h"
065 +#include "proc.h"
066 +
067 +int
068 +sys_factorial(void)
069 +{
070 +    int start;
071 +    if(argint(0, &start) < 0)
072 +        return -1;
073 +    return factorial(start);
074 +}
075 +
076 +int
```

```
+factorial(int start)
+{
+   int ret;
+
+   for(ret = 1; start > 1; start--)
+       ret *= start;
+   return ret;
+}
diff --git a/factorialtest.c b/factorialtest.c
new file mode 100644
index 0000000..3a932ee
--- /dev/null
+++ b/factorialtest.c
@@ -0,0 +1,14 @@
+#include "types.h"
+#include "stat.h"
+#include "user.h"
+
+int
+main(int argc, char *argv[])
+{
+  int fac, base;
+
+  base = 10;
+  fac = factorial(base);
+  printf(1, "factrial %d is %d\n", base, fac);
+  exit();
+}
diff --git a/syscall.c b/syscall.c
index 0918da7..84be706 100644
--- a/syscall.c
+++ b/syscall.c
@@ -98,6 +98,7 @@ extern int sys_unlink(void);
 extern int sys_wait(void);
 extern int sys_write(void);
 extern int sys_uptime(void);
+extern int sys_factorial(void);

 static int (*syscalls[])(void) = {
 [SYS_fork]    sys_fork,
@@ -121,6 +122,7 @@ static int (*syscalls[])(void) = {
 [SYS_link]    sys_link,
 [SYS_mkdir]   sys_mkdir,
 [SYS_close]   sys_close,
+[SYS_factorial] sys_factorial,
 };

 void
diff --git a/syscall.h b/syscall.h
index 59a4576..1fb75d7 100644
--- a/syscall.h
+++ b/syscall.h
@@ -21,3 +21,5 @@
 #define SYS_link   19
 #define SYS_mkdir  20
 #define SYS_close  21
+
+#define SYS_factorial 22
diff --git a/usys.S b/usys.S
index 8bfd8a1..62baf0c 100644
--- a/usys.S
+++ b/usys.S
@@ -29,3 +29,4 @@ SYSCALL(getpid)
 SYSCALL(sbrk)
```

```
141  SYSCALL(sleep)
142  SYSCALL(uptime)
143 +SYSCALL(factorial)
```

3. について

力尽きました…

いやまぁ構造はディスクドライバとほぼ同じになるはずですが、ちゃんと動くものを作るのは（それを確認するためのコードも含めて）かなり大変そうです。

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/3/8 木曜日 [http://peta.okechan.net/blog/archives/1408] |