pzhzqt / **xv6-public**
forked from mit-pdos/xv6-public

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base: **c01c4be6337248b2cf10c5ebd...** ← compare: **master**

2 commits | **9** files changed | **0** commit comments | **1** contributor

Commits on Feb 28, 2018

pzhzqt	Added syscall.	b3eda3f
pzhzqt	removed unimportant code	e00d3cf

Showing **9 changed files** with **41 additions** and **0 deletions**.

Unified | Split

2 Makefile

```
        _usertests\
        _wc\
        _zombie\

 fs.img: mkfs README $(UPROGS)
        ./mkfs fs.img README $(UPROGS)
        mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c
kill.c\
        ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c
zombie.c\
        printf.c umalloc.c\
        README dot-bochsrc *.pl toc.* runoff runoff1
runoff.list\
        .gdbinit.tmpl gdbutil\
```

```
        _usertests\
        _wc\
        _zombie\
+       _test\

 fs.img: mkfs README $(UPROGS)
        ./mkfs fs.img README $(UPROGS)
        mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c
kill.c\
        ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c
zombie.c\
        printf.c umalloc.c\
+       test.c\
        README dot-bochsrc *.pl toc.* runoff runoff1
runoff.list\
        .gdbinit.tmpl gdbutil\
```

5 proc.c

```
   memset(p->context, 0, sizeof *p->context);
   p->context->eip = (uint)forkret;

   return p;
 }
```

```
   memset(p->context, 0, sizeof *p->context);
   p->context->eip = (uint)forkret;

+  int i;
+  for(i=1;i<23;i++){
+         p->cnt[i]=0;
+  }
+
   return p;
 }
```

2 proc.h

```
@@ -1,3 +1,4 @@

 // Per-CPU state
 struct cpu {
   uchar apicid;                // Local APIC ID
```

```
+#include "syscall.h"
 // Per-CPU state
 struct cpu {
   uchar apicid;                // Local APIC ID
```

```
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
};

// Process memory is laid out contiguously, low addresses
first:
```

```
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
+  int cnt[23];                 // count table of system
calls
};

// Process memory is laid out contiguously, low addresses
first:
```

3 🟩🟩🟩⬜⬜ syscall.c

```
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
};

void

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
```

```
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);
+extern int sys_count(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
+[SYS_getCallCount] sys_count,
};

void

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
+       curproc->cnt[num]++;
    curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
```

1 🟩⬜⬜⬜⬜ syscall.h

```
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
```

```
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
+#define SYS_getCallCount  22
```

13 🟩🟩🟩🟩🟩 sysproc.c

```
  release(&tickslock);
  return xticks;
}
```

```
  release(&tickslock);
  return xticks;
}
+
+int
+sys_count(void)
+{
+       int n;
+       if(argint(0,&n)<0){
+               return -1;
+       }else if(n<1||n>22){
+               return -1;
+       }else{
+               return myproc()->cnt[n];
+       }
+}
```

13 🟩🟩🟩🟩🟩 test.c

```
...  @@ -0,0 +1,13 @@
+#include "types.h"
+#include "user.h"
+#include "fcntl.h"
+#include "syscall.h"
+
+int main(void){
+       if (fork()==0){
+               printf(1,"fork called %d times in child process\n",getCallCount(SYS_fork));
+       }else{
+               printf(1,"fork called %d times in parent process\n",getCallCount(SYS_fork));
+       }
+       exit();
+}
```

1 🟩⬜⬜⬜⬜ user.h

```
 char* sbrk(int);                     char* sbrk(int);
 int sleep(int);                      int sleep(int);
 int uptime(void);                    int uptime(void);
                                     +int getCallCount(int);

 // ulib.c                            // ulib.c
 int stat(char*, struct stat*);       int stat(char*, struct stat*);
```

1 🟩⬜⬜⬜⬜ usys.S

```
 SYSCALL(sbrk)                        SYSCALL(sbrk)
 SYSCALL(sleep)                       SYSCALL(sleep)
 SYSCALL(uptime)                      SYSCALL(uptime)
                                     +SYSCALL(getCallCount)
```

No commit comments for this range