如何写自己的测试进程

Lab4引入系统调用后，无法通过改动init.c来测试功能，需要编写自己的用户进程。这里简单介绍一下如何写一个自己的测试进程。

1. 编写程序

在`user`目录下新建一个名为`test.c`的文件，这个文件的框架是：

```
#include "lib.h"
void umain()
{
    writef("hello, world!\n");
}
```

`umain`中可以编写你自己的测试函数。

2.修改Makefile

在`user/makefile`中的`all`规则中添加`test.x test.b`

```
all: test.x test.b fktest.x fktest.b pingpong.x pingpon
```

3. 修改init.c

在init.c中的合适位置添加

```
ENV_CREATE(user_test);
```

随后编译运行，即可观察到测试进程的输出信息。

```
hello, world!
[00000800] destroying 00000800
[00000800] free env 00000800
i am killed ...
```

附：一个用户进程的生命周期

用户进程的入口点在`user/entry.S`中定义如下：

```
_start:
    lw  a0, 0(sp)
    lw  a1, 4(sp)
    nop
    jal libmain
    nop
```

可以看出，用户进程将跳转到libmain函数，这个函数在`user/libos.c`中定义：

```
void libmain(int argc, char **argv)
{
    // set env to point at our env structure in envs[].
    env = 0;    // Your code here.
    int envid;
    envid = syscall_getenvid();
    //通过系统调用获得envid
    envid = ENVX(envid);
    env = &envs[envid];
    /*
        envs的地址为0x7f400000，这里之所以可以在用户态使用envs数组，
        是因为在mips_setup_vm中，我们通过
        boot_map_segment(pgdir, UENVS, n, PADDR(envs), PTE_R);
        进行了映射
    */
    // call user main routine
    umain(argc, argv);
    //umain就是我们编写的用户进程的主要部分
    // exit gracefull
    exit();
    //exit函数通过系统调用syscall_env_destroy(0)，杀死本进程
}
```

以上即为一个用户进程从入口到死亡的生命周期。

AUTHOR:WYK