Introduction to Computer Science

Version 2018

PA4

作者: 郑奘巍

学号: 171860658

邮箱: zzw@smail.nju.edu.cn

专业: 计算机科学与技术系

December 20, 2018

1 进程报告

我完成了所有的必做内容和大量选做内容，包括让 DiffTest 支持分页机制，支持开机菜单程序的运行, 实现 Navy-apps 上的 AM 等。

由于本机 F1 键有问题，最后在 NEMU 中实现分页机制的实验中，切换程序使用的是左中括号键 [，右中括号键]，反斜杠\三键，默认加载程序为后台 hello，前台为开机菜单程序共三个。请老师查收时注意，谢谢。

2 必做题

Problem:

分时多任务的具体过程: 请结合代码, 解释分页机制和硬件中断是如何支撑仙剑奇侠传和 hello 程序在我们的计算机系统 (Nanos-lite, AM, NEMU) 中分时运行的.

Solution:

我试图通过分析整个仙剑奇侠传的加载运行的过程来回答这个问题。

在加载之前，`_vme_init()` 会为我们设置内核的页表，并让 NEMU 设置好 cr0,cr3. 首先，在 loader 中我们加载仙剑奇侠传的源程序，计算其所需要的页面并通过调用 `new_page()`,`_map()` 函数进行分配。在分配完后设置堆区的起始位置。

上述的加载发生在 Nanos-lite 中，调用了属于 AM 的 `_map()` 。该函数将把信息注册到对应页目录对应页表的页表项中。当出现页表缺页时，会申请新的页表。

如此一来，仙剑奇侠传便通过分页机制加载成功。当其需要申请堆区空间时，将通过 `mm_brk()` 申请空间。在访问地址时，由于分页机制访问的是虚拟地址，将通过 NEMU 中 `vaddr_read()` 和 `vaddr_write()` 将虚拟地址转化到相应的物理地址。

在仙剑运行时，将会收到来自时钟的硬件中断。当收到中断后，通过一系列跳转（类似于 PA3 中的系统通用，不再详述），从 AM 的 `irq_handle()` 函数中进入 Nanos-lite 的 `schedule()`，选择新的程序。而在 `irq_handle()` 中，我们将保存旧上下文，设置新的上下文（在 NEMU 中设置 cr3), 从而启动 hello 程序。当条件成熟，仙剑奇侠传会在时钟中断中被重新启动。

---

3 选做题

Problem (1):

尝试通过`context_kload()` 来加载在 Nanos-lite 中定义的`hello_fun()` 函数, 来替换 hello 用户进程, 你应该会观察到缺页错误. 尝试定位并修复这个问题.

Solution:

`hello_fun()` 运行在内核态，当切换回去时需设置相应的上下文。下面的代码是我的实现，kbase 是内核态的 `_Protect` 指针。

```
extern _Protect* kbase;
_Context *_kcontext(_Area stack, void (*entry)(void *), void *arg) {
    _Context *c = (_Context*)stack.end - 1;
    c->eip= (intptr_t)entry;
    c->cs = 8;
    c->prot = kbase;
    *((uintptr_t *)stack.start) = (uintptr_t)c;
    return c;
}
```

Problem (2):

让 DiffTest 支持分页机制

Solution:

在 `restart()` 函数中初始化 CR0，在 `vaddr_read` 和 `vaddr_write` 中根据 i386 手册对于 accessed 位和 dirty 位的描述更新这两位，并在初始时给这两位赋值 0。具体实现主要使用了如下两个宏：

```
#define SET_DIRTY(x) (is_write?((x)|0x40):(x))
#define SET_ACCESS(x) ((x)|0x20)

```

Problem (3):

支持开机菜单程序的运行

Solution:

为支持开机菜单程序的运行，需要在 `SYS_execve` 系统调用时加载相应的程序，并返回当前的 `_Context`。同理，在 `SYS_exit` 时重新加载开机菜单程序。`SYS_execve` 的实现如下：

```
case SYS_execve:
    context_uload(&pcbbase[proc_cur_select], (char *)a[1]);
    c->GPRx = 0;
    return pcbbase[proc_cur_select].cp;
```

Problem (4):

实现 Navy-apps 上的 AM

Solution:

为实现Navy-apps上的AM,我们需要实现`trm.c`,`ioe.c`,`device/timer.c`,`device/input.c`,de通过调用`ndl.c`中封装好的函数，仿照`x86-nemu`的设计，即可完成Navy-apps的设计。以上文件的主要对应函数如下表，具体代码请见`x86-navy`.

trm.c	putchar()/exit()
input.c	NDL_WaitEvent
timer.c	NDL_WaitEvent
video.c/vag_init	NDL_OpenDisplay
video.c/video_write	NDL_DrawRect/NDL_Render
video.c/video_read	-