

Contents

1. 2016年操作系统课程设计-v9-cpu上的ucore移植-基于Node.js的V9 CPU模拟器和调试器

1. 已有的相关工作

2. 日志

1.

1. 2016年5月29日 第十五周 周日

2. 2016年5月26日 第十四周 周四

3. 2016年5月20日 第十三周 周五

4. 2016年5月15日 第十三周 周日

5. 2016年5月3日 第十一周 周二

6. 2016年5月2日 第十一周 周一

7. 2016年5月1日 第十一周 周日

(周日是一周之始)

8. 2016年4月30日 新建虚拟机

oslab-v9

9. 2016年4月30日 第十周 周六

10. 2016年4月29日 第十周 周五

11. 2016年4月27日 第十周 周三

12. 2016年4月25日 第十周 周一

13. 2016年4月23日 第九周 周六

14. 2016年4月18日 第九周 周一

15. 2016年4月16日 第八周 周六

16. 2016年4 月15日 第八周 周五

2016年操作系统课程设计-v9-cpu上的ucore移植-基于Node.js的V9 CPU模拟器和调试器

在本页面维护专题训练大实验"v9-cpu上的ucore移植:基于Node.js的V9 CPU模拟器和调试器"的相关信息。老师、助教和选做课程设计的同学可修改该页面的内容。

实验目标描述：在本地和网络浏览器环境中，设计并实现基于v9 cpu的在线ucore实验环境中的“基于Node.js的V9 CPU模拟器和调试器”部分。可能的工作内容如下。

1. ；

实验参与者信息

姓名	学号	电子邮箱	GitHub 账户
马			

坚 鑫	2013011384	 majx13fromthu@gmail.com	 https://github.com/JianxinMa
--------	------------	----------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------

实验代码仓库：


-  <https://github.com/JianxinMa/v9.js>

已有的相关工作


- [[]]

日志

2016年5月29日 第十五周 周日

最终报告文档： <https://github.com/JianxinMa/v9.js/blob/gh-pages/README.md>

调试器设计单独列在另一文档（README.md中也有给出此链接）：
<https://github.com/JianxinMa/v9.js/blob/gh-pages/doc/debugger.md>

最终报告幻灯片： https://github.com/JianxinMa/v9.js/raw/gh-pages/doc/final_20160529.pptx

2016年5月26日 第十四周 周四

Emscripten翻译的编译器、镜像生成工具看来是比较稳定了。


现在增加了JavaScript版本的C预处理器。开始结合ucore组的移植成果进行测试。

2016年5月20日 第十三周 周五

完成编译器、磁盘映像生成工具的移植（从C到JS）。协助工具链小组在模拟器上增加了alex指令集。

接下来需要：（1）协助工具链小组增加服务端、统一调试信息的格式；（2）集成移植小组的ucore，需增加c preprocessor（调试信息的生成过程需要调整）。

2016年5月15日 第十三周 周日

报告：

https://github.com/JianxinMa/v9.js/raw/master/doc/semifinal_20160515.pptx

注：为了需要跑起ucore小组移植的ucore，我还需要尽快实现一个浏览器端的C预处理器（只需支持include和define）。

2016年5月3日 第十一周 周二

现在可以通过<http://166.111.68.197:11293/> 访问demo网站了。

今早发现之前写的服务端处理用户修改源代码的方式not thread-safe，故把这个功能关了。所以用户如果修改源代码不会有效果。

2016年5月2日 第十一周 周一

用一个树状图展示出所有变量的值，完成。之所以用树状图，是为了方便地展示出数组（从一个数组伸出它的各个元素）、结构体（从一个结构体伸出它的各个成员）、甚至结构体为元素的数组、数组的数组，等复杂结构。用户可以通过点击结点把子节点展开或收起来。

不足之处：（1）有些变量会因缺页而看不到值，虽是正常现象，但不方便，需解决；（2）需给用户提供查看任意地址（由用户给出类型）的接口；（3）现在的界面不太友好，比如当一个数组的元素很多的时候它们会挤在一起，导致看不清楚。

2016年5月1日 第十一周 周日（周日是一周之始）

关于调试器的细节五：

- 对于os.c的一条指令，应该利用实址还是虚址去定位它在源代码中的位置？答：除了init process对应的那几个函数，其它都应该使用实址。因为一开始载入os.c的可执行文件时没开paging，且可能有多个虚址实际指向同一个实址。
- 对于用户程序的一条指令，用实址还是虚址？答：虚址。因为载入时paging已开。

细节六：

- 编译器编译的时候认为.text是从0x0开始，生成的编译信息也这样认为。启动时os.c的.text确实在0x0，但由于os的exec实现把类elf文件的hdr放在了0x0，它占了16bytes，所以通过exec启动的程序.text是在0x10开始。因此调试器这时需要加偏移值0x10。
- 同上，.data，.bss也需要相应偏移。

细节七：

- 之前提到过，v9处理main函数return的方式和linux不一样，v9里是由os在exec的时候把main的返回地址设置为指向一条trap S_exit的指令。但这条指令是存在在栈上的，这很不规范，而且导致我们的调试器一旦遇到main函数return就必然会跟丢，因为我们编译器生成的调试信息是绝对没有考虑到

这条构造出来的trap S_exit。

- 解决方案：特殊判断，如果接下来要执行的指令的地址在栈，且内容是trap S_exit，就放行。

解决了这些细节后，似乎从os到init到sh到ls、cat、hello等各种子程序的跟踪都比较正常了。

正在检查是否还有明显错误，并优化用户体验，力争尽快给出一个适合公开的版本。

另外，编译器生成的位置信息还是不太准：比如"for (i = 0; i < 1000; i++) i = i + 1 - 1;\n return 0;"这个例子，for循环结尾的跳转对应的汇编指令居然被归到“return 0;”去了，会给用户造成很大困惑。

2016年4月30日 新建虚拟机oslab-v9

新建虚拟机oslab-v9的访问方法：

```
ssh -p 11291 xyong@166.111.68.197
http://166.111.68.197:11293
```

用户名：s2013011384

2016年4月30日 第十周 周六

为了保证调试器的正确性，在这里简要描述下os.c的进程、特权级切换流程：

1. 刚启动，OS首先在内核态依次设置内核页表、终端、中断向量、磁盘缓冲、磁盘、时钟；
2. 仍然在内核态，构造第一个用户进程，它后面会执行程序是root/etc/init.c，但现在还没执行它，先输出“Welcome!”到终端，然后再调用scheduler()；
3. 仍在内核态，scheduler选择好进程后（第一个肯定是init.c），做了两件重要的事：一是设置页表为进程的页表；二是切换context（表现为切换栈）；
4. scheduler切换栈这个动作很重要，因为每一个进程初始时在栈上把函数返回地址设置为了forkret()这个函数。这样，一旦切换到一个新进程的context，接下来函数返回指令LEV就会使它进入forkret，而在forkret中最后会执行RTI（中断返回），从而进入用户态，并进入用户程序。

上面的流程告诉我们，不能再设置PDIR的时候就切换符号表到init.c的符号表，因为os.c中还有一些没执行完。正确的做法是在RTI后才切换符号表。

这里补充细节四：

- init process并不是只执行了init.c的程序，它一开始执行的是在os.c的一小段用户态程序，由这段程序通过exec调用执行init.c。
- 这意味着一开始.text会被覆盖成os.c中的那一段用户态程序；

- 直到那一小段用户程序exec了init.c，.text才会被覆盖成init.c的内容；
- 重要启示：RTI后进入os.c的那一小段用户程序时，如果尝试载入符号表，会发现没有符号信息在.text！这时仅仅fallback到os.c的符号信息是不够的，因为所有的地址会发生偏移，比如说，这段小程序的init_start函数原本是在os.c的0XXXXXXXXX处，但现在偏移到了0x0（.text开头），故需要特殊处理。

那么现在由os.c进入到了init.c了，同时也切换到了用户态，接下来：

- 经过一翻设置后，init.c这个init process就fork了一下，在fork出来的孩子里exec了/etc/bin/sh.c，而原来的init process进入wait。

值得注意的是init.c的时候已经开始有系统调用了，需要注意在TRAP的时候把符号表切换为os.c的，再在RTI时换回正在执行的进程。在RTI返回时切换符号表的另一个好处是这种处理把exec的情况也囊括了。

那么接下来就主要是sh.c在干活了：

- sh.c基本上就是不断读键盘输入，不断fork再exec子程序，这点和init.c本质上没有什么差别。

关于程序退出：

- 如果程序是通过exit系统调用退出的，那么在TRAP载入os.c的符号表，再在RTI时切换回用户程序的符号表应该也是正确的：在TRAP到RTI这期间，os.c主要做了exec，而exec的末尾会执行sched，sched才RTI到其他用户程序。
- 但如果程序是因为main里return而退出的呢？在linux里main返回后是到_start中，由它负责exit。那么v9呢？v9的解决方案是让os在exec时设置新进程的main的返回地址，使其指向一条S_exit系统调用，对，它是在os.c而不是c.c干这件事。

此外，还有外设中断导致的切换！和trap一起统一处理。

2016年4月29日 第十周 周五

之前采用“模拟器在设置PDIR时切换和当前程序对应的符号表”的方案有不少问题。

细节一：

- 载入符号表时访存可能缺页，此时需发出缺页异常，接下来必跳到缺页处理例程。这个时候要fallback到os.c的符号表，以继续跟踪。
- 根据RTI可判断缺页处理是否完成，此时应重新尝试载入符号表。

细节二：

- 用户程序间调度需要换PDIR，但存在其他设置PDIR的事件，如sbrk。因此仅凭设置PDIR这个事件就断然去载入符号表是不充分的。

- 用户程序通过trap切换到内核，并不需要切换PDIR，皆因os.c的exec实现会先把内核页表的内容拷贝到用户程序的页表！故凭PDIR去载入符号表也非必要条件。

细节三：

- 正如细节一所说，载入符号表可能需要处理缺页。如果用户正在单步调试、或设了断点，他/她就会发现程序跑入了异常处理处，但非调试时此时应该不会进入异常处理才是，这可能会使用户困惑。

需要继续仔细阅读“os.c”更加全面地理解各个程序及特权级的切换流程。需要用更好的方式触发切换符号表的动作。

本周截止目前完成：（1）编译器所有所需调试信息获取完毕；（2）一个功能更丰富的服务器端；（3）给模拟器添加显示变量值功能时发现上述种种问题，着手修正，正在进行；（4）refactor之前赶进度留下的丑陋代码，修正逻辑控制的潜在问题。

三天内的首要短期目标：搭建demo环境，以收集用户反馈。

2016年4月27日 第十周 周三

因为希望能够实现一个不需要服务端的版本，所以尝试使用Emscripten等工具将编译器译成JavaScript版本，但是发现：

- Emscripten产生的js代码十分巨大，数万行，且默认是在node.js环境下的，需要手动改成适合浏览器的，但是产生的代码不适合手动修改。
- 尝试使用一种含C语言特性的JavaScript方言，LLJS，希望能简化移植到js的工作量，但LLJS已在三年前废弃。
- bonsai-c号称希望产生比Emscripten更可读的C代码，但同样很久没人维护，且很多C特性都不支持。

总结：如果想摆脱服务器端，似乎只能**手动重写**js版的编译器。考虑到时间，算了，我还是乖乖写服务器端去吧。然后让用户自己跑一个local的服务器？但需要尽量保证服务器端容易搭建（windows搭环境比较麻烦）、且必须同时支持linux、osx、windows（windows比较特殊），不然用户一定会骂我的。如果后面时间不紧张才考虑用JS手写编译器。

2016年4月25日 第十周 周一

完成变量的类型信息（含复杂的结构体）的获取。接下来还需要在模拟器增加相应支持。

2016年4月23日 第九周 周六

本周完成了（我承认本周进度比上周慢了些）：

- 基本摸清了v9现有的编译器c.c的实现（读几千行代码真的好心累）。
- 从编译器获得了局部变量、全局变量的地址信息。
- 能够获得简单的类型信息（包括指针、指针的指针...），但是结构体还不行，数组能知道是数组，但不知道其基本类型（已解决）。

还需要尽快完成：

- 对c.c的类型系统进行更深入的理解，尤其细节处，以获得复杂变量的类型信息。

4月24日 补充：

- 向老师开会时提到c9这个在线IDE（见<https://github.com/c9/core>）。我去了解了下，认为在我们项目里采用这个东西是可行的，但是应该比较花时间，故暂时不打算用这个东西。后来人若有兴趣可以尝试之。

2016年4月18日 第九周 周一

修正了4月15-16日的问题，原来是因为：

- 我是在PDIR时触发载入符号表事件；
- 但是OS在sbrk的时候也会触发PDIR，这时候的0x10处不是我想要的内容。

后面还需要再细看OS，以检查是否有其他corner cases。

现在从原理上验证了目前的debugger设计是可以work的，但是这还是一个prototype，还需要解决：

- 不少程序需要用户输入，断点、单步调试时应该有相应的提示才不会让人茫然无措；
- 目前编译器生成的信息还不够准；
- 设置断点后有时候会重复停在断点多次；
- 有些地方设置断点是无效的，但目前没有提示。

下一个工作量较大的目标是：

- 查看变量的值：主要工作量在编译器（然而我为了验证想法还是需要自己改c.c）、模拟器、前端。

2016年4月16日 第八周 周六

针对切换到用户程序时载入符号表不正确的问题，目前的分析：

- OS的.text载入到虚址0x0，但用户程序的.text载入到虚址0x10，多了16bytes（这16bytes是程序的header）。但修正这个后仍有问题。
- 奇怪的是，在用户程序中printf能看到0x10处有符号信息，但在模拟器中把内存（剔除用来模拟硬盘的部分）全部扫了一趟却发现没有符号信息.....

备注：我把调试需要的信息放在了可执行二进制文件的.text段开头（紧跟在16bytes的header之后）。

2016年4月15日 第八周 周五

经过一整周的日夜奋战，完成如下事项：

- 一个在线平台，有编辑器（可切换、编辑多个文件）、模拟器、调试功能等等。
- 修改了c.c，输出符号信息。
- 支持单步调试，可定位到代码的具体哪一行。
- 支持设置断点进行调试，可同时在多份代码中设置断点（但目前从OS转至用户程序后尚有问题）。
- 上述调试功能在打开页机制后仍能够正常工作。
- Github项目的README.md中给出了demo网址，但如果要运行模拟器，还需要在本地额外运行一个迷你服务器。

如果需要演示，欢迎电邮我。

目前仍有一些挺麻烦的问题要解决：

- 运行过程中，OS执行用户程序（比如OS执行/etc/init）后，调试器随之切换到用户程序，但在获取符号信息时出现问题。
- 自己改的c.c输出的符号信息在一些地方不够准确，相差一两行，且似乎少输出了一些信息。
- UI还欠打磨，用户如果乱点估计会出我意料之外的问题。

从本周的开发感受来看，解决上述问题预计会在下一周消耗我较多精力。但仍希望下周能有时间增加输出变量内容的功能。

OS2016spring/projects/u9/emulator (last edited 2016-05-29 15:02:09 by 2013011384)

MoinMoin Appliance - Powered by TurnKey Linux