**实验题目：ELF场景扩展**

# 实验内容：

1. 使用Unikraft成功运行elfloader程序

2. 动态reload连接

3. 头脑风暴

# 实验过程：

## 使用Unikraft成功运行elfloader程序

和前面的构建过程基本一致。

首先安装好相关的依赖库。可以通过以下命令安装app-elfloader本体、lib-lwip库以及lib-libelf库。

|  |
| --- |
| git clone https://github.com/unikraft/app-elfloader/  cd app-elfloader/  git clone https://github.com/unikraft/lib-lwip .unikraft/libs/lwip  git clone https://github.com/unikraft/lib-libelf .unikraft/libs/libelf |

需要注意修改一下Makefile文件中unikraft和libs的路径问题，这里不多赘述。

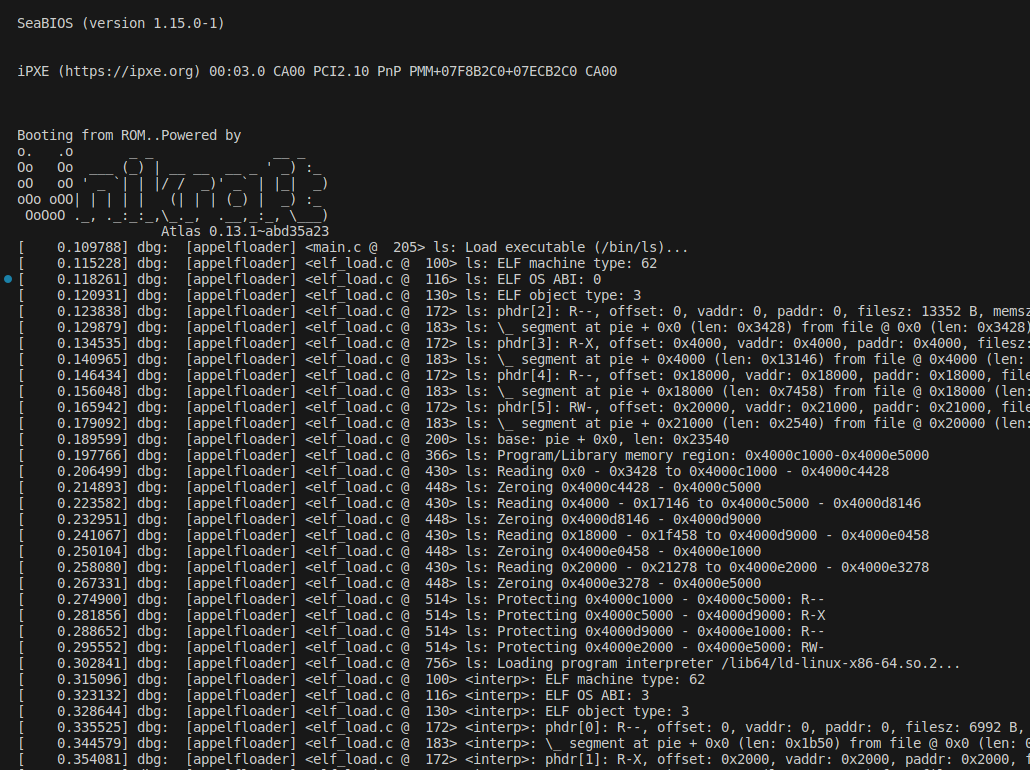
然后我们根据项目给出的命令对unikraft进行配置。

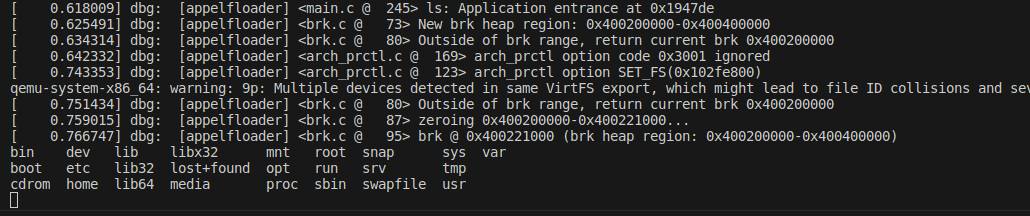
|  |
| --- |
| UK\_DEFCONFIG=$(pwd)/.config.elfloader\_qemu-x86\_64 make defconfig  make -j $(nproc) |

通过以上命令，就可以得到最终的镜像文件了。我们通过下面的命令对其进行运行。该命令将 /bin/ls作为附加信息传递给了qemu，在elfloader程序中，会根据该目录对该程序进行加载。

|  |
| --- |
| sudo /usr/bin/qemu-system-x86\_64 \      -fsdev local,id=myid,path="/",security\_model=none \      -device virtio-9p-pci,fsdev=myid,mount\_tag=fs0,disable-modern=on,disable-legacy=off \      -kernel build/elfloader\_qemu-x86\_64 -nographic \      -enable-kvm -cpu host \      -append /bin/ls |

可以得到运行的结果如下图所示。成功运行了/bin/ls二进制程序，将根目录下的内容都打印出来了。





观察源码可以指导elfloader实现动态构建的函数是elf\_load\_img和elf\_load\_vfs，即下面的部分。

|  |
| --- |
| #if CONFIG\_APPELFLOADER\_INITRDEXEC      uk\_pr\_debug("%s: Load executable...\n", progname);      prog = elf\_load\_img(uk\_alloc\_get\_default(), (void \*) img->vbase,                  img->len, progname);  #else /\* CONFIG\_APPELFLOADER\_VFSEXEC \*/      uk\_pr\_debug("%s: Load executable (%s)...\n", progname, path);      prog = elf\_load\_vfs(uk\_alloc\_get\_default(), path, progname);  #endif /\* CONFIG\_APPELFLOADER\_VFSEXEC \*/ |

在实际使用中，我们只需要令unikraft可以接收到要动态构建的程序的路径，就可以将其通过elf\_load\_vfs及相关代码，实现动态创建了。

可以类比baby-step部分的echoback，使unikraft可以接收指令，以获得二进制文件程序的路径。

## 头脑风暴

Elfloader提供的动态构建的功能有很多意义。

比如可以节省内存空间，令程序可以在需要的时候才加载共享库或插件等模块；

提高运行效率，令程序可以不要在加载一次性花太多时间；

可以动态更新更加灵活，令程序可以在运行时直接动态地更新共享库或插件等模块，而无需重新启动整个程序。；