

Ource OS说明文档

《计算机操作系统》学期作业

|  |  |
| --- | --- |
| 姓 名 | 许祖耀 |
| 学 号 | 2107010120 |
| 专业班级 | 计算2101 |
| 学 院 | 计算机科学与技术学院 |

2024年5月27日

# 开发环境

Ource OS参考川合秀实编著，周自恒翻译的《30天自制操作系统》中的Haribote系统编写，原书的附带资源中提供了一系列开发工具，Ource OS正是通过原书附带的开发工具编写而成的。

原书提供的开发工具包括但不限于以下内容：

* GNU Make – GNU项目中的工具之一，用于自动化编译和链接程序。
* Nask - NASM开源汇编器，支持 Intel 语法，将.nas文件编译为.obj文件
* cc1 - GCC的一个组件，负责将C源代码编译为AT&T语法的.gas汇编文件。
* gas2nask – 作者者编写的工具，用于将GNU 的汇编代码转换为NASM汇编代码。
* obj2bim – 作者编写的链接工具,可以将对象文件转换为.bim二进制映像文件。
* bim2hrb – 作者提供的工具,可以将.bim二进制映像文件转换为HariboteOS 中的.hrb可执行二进制文件。
* edimg – 用于创建和编辑磁盘映像文件。
* makefont – 可以将字库文件转换为二进制字库文件.
* bin2obj – 将二进制字库文件转换为语言数组,并储存在.obj文件中。
* QEMU - 通用开源的计算机仿真器和虚拟器。

在《30天自制操作系统》这本书的上下文中，作者为了教学与实验目的,使用了自创的.hrb文件格式作为书中Haribote操作系统的可执行二进制文件的目标格式, Ource 也因此将其沿袭了下来。

Ource是通过利用上述开发工具, 效仿书中Haribote OS的开发方式而编写的.

# 功能介绍

Ource OS实现了操作系统内核程序应该提供的支撑功能和资源管理功能。

## 支撑功能

### 中断处理

通过初始化可编程中断控制器，设置相应的中断向量表，Ource实现了对中断的处理，目前支持的中断信号有：

* 异常中断 中断信号0x0d
* 时钟中断 中断信号0x20
* 键盘中断 中断信号0x21
* 鼠标中断 中断信号0x2c（支持但未使用）
* 电气中断 中断信号0x27
* 系统调用 中断信号0x40

### 时钟管理

通过初始化可编程中断计时器PIC，定义计时器数据结构，实现时钟中断服务程序，Ource完成了时钟管理功能。

### 原语操作

通过汇编语言编写的\_io\_cli\_和\_io\_sti\_函数设置PIC的中断屏蔽位，Ource OS实现了关中断和开中断功能，以执行相应的原语操作。

## 资源管理

Ource充分担当了OS内核应具备的资源管理者身份，实现了进程管理、内存管理、设备管理、文件管理功能。

### 进程管理

Ource是一个多道分时操作系统，每一个运行在其上的程序都一一对应着一个TCB任务控制块（类似PCB），Ource使用多级优先队列调度算法实现任务的调度，采用时间片轮转的方式进行任务切换，具有高优先级的任务将会被分配到更长的时间片。

### 内存管理

Ource使用空闲分区链表记录和追踪内存中的空闲分区使用情况，并采用分段存储管理的方式进行内存管理，通过设置全局段描述符表GDT和局部段描述符表LDT，将内核程序和不同的用户程序分配在具有不同权限的内存段中，以进行内存的管理。

### 设备管理

Ource通过数个简单的.c代码文件，分别与键盘、屏幕等设备控制器进行直接通信，实现设备的驱动和管理功能。

### 文件管理

Ource使用FAT12文件系统进行文件管理，目前仅支持文件系统中的文件读取操作。

## 更多细节

Ource以《30天自制操作系统中的》中作者提出的.hrb文件格式作为系统的可执行文件，并使用其提供的工具集进行程序代码编译和链接。

Ource通过设置不同程序的内存段属性，实现操作系统和应用程序之间的隔离与保护。Ource向用户提供了联机命令接口，用户可以通过在命令行窗口中输入命令进行操作；同时Ource也提供程序接口，用户程序可以通过系统调用函数，发出中断信号，转移到相应的服务例程中，目前提供了往控制台终端输出字符和换行的系统调用函数api\_putchar和api\_end。

Ource的屏幕显示模式默认为1024x768x8bit彩色，并通过调色板模式显示彩色，通过修改内存中VRAM指定的区域，Ource控制屏幕中每个像素的显示色彩，并通过外部挂载的二进制字库文件逐像素绘制字符。

# 模块架构

Ource OS的内核功能由14个功能模块组成,其分别为.c文件或.nas汇编语言文件分别负责各个独立功能,共同组成Ource的核心内核程序. Ource OS的核心功能文件如下:

* ipl10.nas – 汇编语言编写的操作系统引导扇区IPL程序, 负责初始化计算机硬件并加载操作系统的其余部分.
* asmhead.nas - 汇编语言编写的内核启动代码，负责检测硬件环境，初始化内核运行环境等。
* naskfunc.nas – 定义了内核程序需要的且只能通过汇编语言实现的辅助函数等，如关中断\_io\_cli和开中断\_io\_sti。
* bootpack.c – C语言编写的内核主程序,负责初始化描述符表、内存、进程等。
* console.c – 命令行终端的相关功能实现，向用户提供命令接口。
* dsctbl.c – 用于管理各种描述表, 如全局段描述符表GDT，中断描述符表IDT。
* fifo.c – 先入先出型缓冲区的相关实现代码。
* file.c – 对FAT12文件系统进行操作与管理的相关代码实现。
* int.c – 负责初始化可编程中断控制器PIC，并实现部分必需的中断服务程序。
* keyboard.c – 负责驱动键盘，与键盘设备控制器进行通信，定义了键盘中断的中断服务程序。
* memory.c – 内存管理部分的代码实现，使用空闲分区链表进行内存分配、回收等，
* mtask.c – 任务（进程）管理的代码实现，使用多级优先队列，采用时间片轮转方式进行进程调度等。
* timer.c 时钟管理部分的代码实现，负责初始化可编程间隔计时器，向应用程序分配计时器等，定义了时钟中断的中断服务程序。
* graphic.c – 负责驱动屏幕，进行图形绘制的相关代码实现。
* window.c – 在屏幕上绘制应用窗口的相关代码。

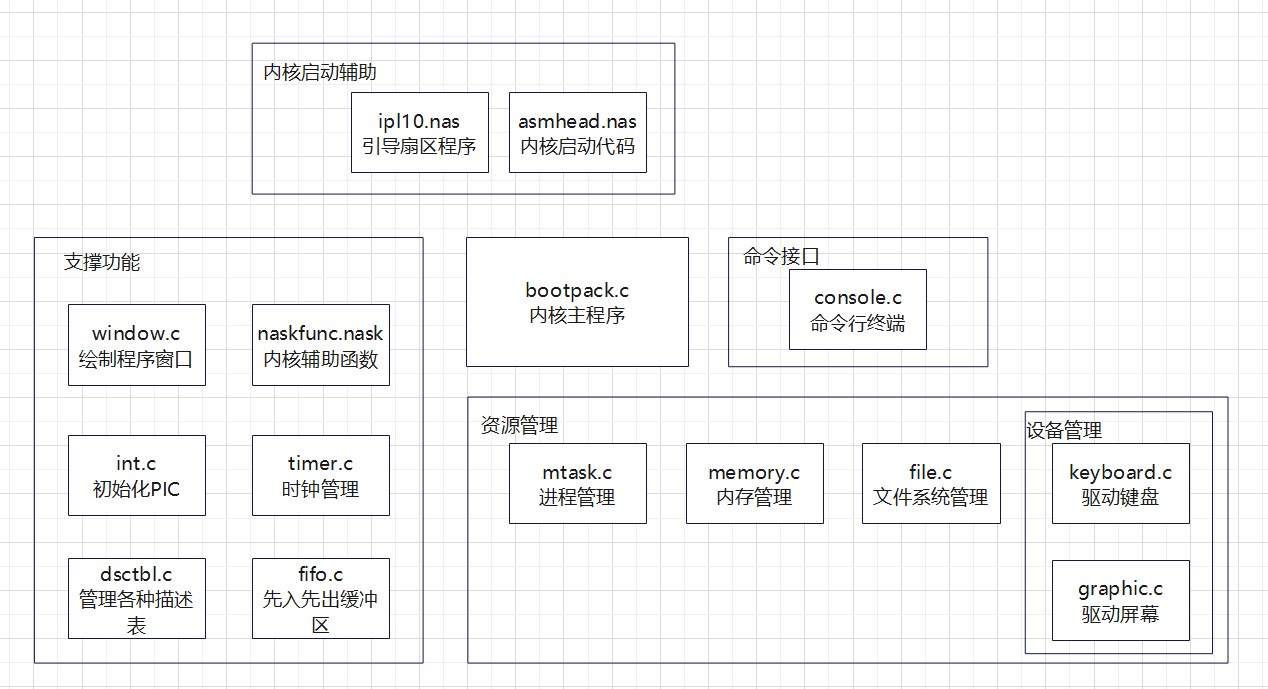


图 1 Ource 内核模块组织结构图

# 运行流程

## 整体工作流程

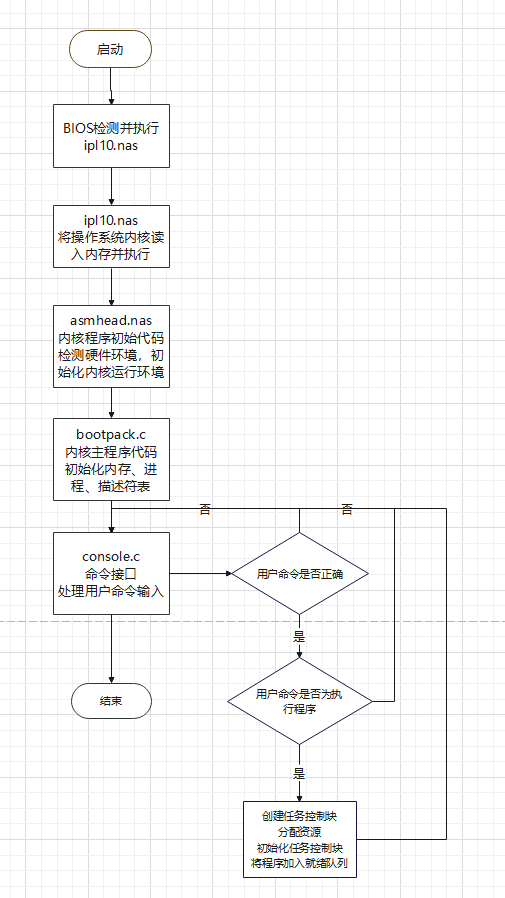


图 2 Ource启动后的工作流程

## 内核主程序bootpack.c工作流程

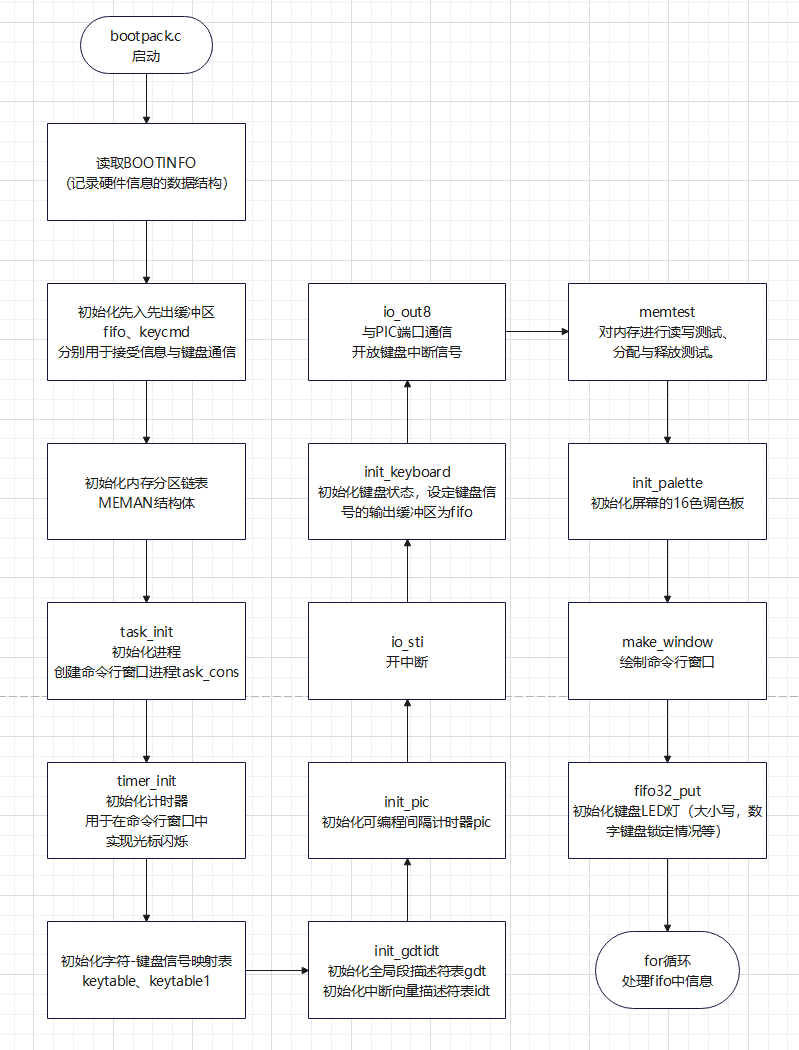


图 3 bootpack.c内核主程序的工作流程

# 详解说明

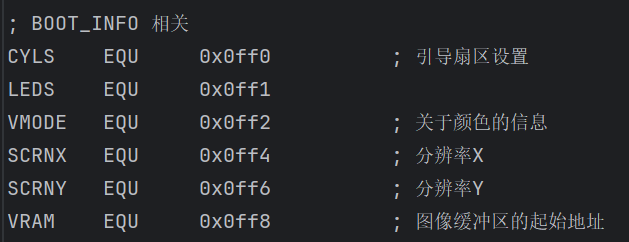
Ource的代码中有着若干个重要的数据结构，其储存了OS内核运行所需要的关键信息。

## BOOTINFO结构体

用于储存计算机的相关硬件信息的结构体。



BOOTINFO结构体定义在asmhead.nas文件中，其位于内存地址单元0xff0处。

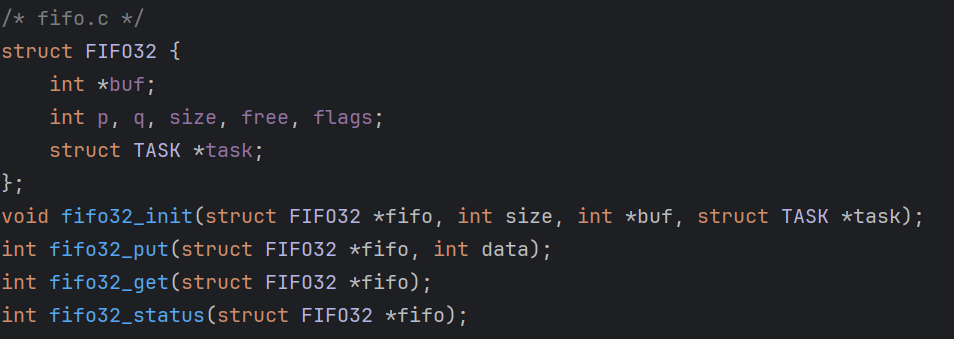


成员变量说明：

* cyls：引导扇区的结束位置
* leds: 启动时键盘LED灯状态
* vmode：显卡的显示模式
* reserve：保留字节
* scrnx，scrny：画面的横纵分辨率
* vram：图像缓冲区的起始位置，通过修改vram可设置屏幕像素的显示颜色。

## FIFO32结构体

实现先入先出的消息队列的数据结构，每个消息大小为4字节，32位，以int型表示。



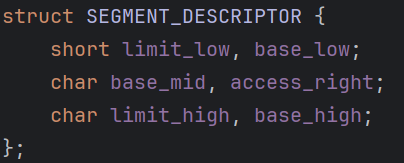
相关功能实现在fifo.c文件中。

成员变量说明：

* buf：指向消息队列的指针
* p：队首指针
* q：队尾指针
* size：消息队列的大小
* free：剩余的空闲空间数
* flags：表示消息队列的状态（为1时表示发生溢出）
* task：指向消息队列所服务的进程

## SEGMENT\_DESCRIPTOR 结构体

段描述符数据结构，用于定义段的属性。



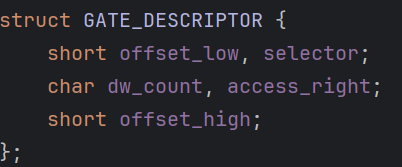
相关功能实现在dsctbl.c文件中。

成员变量说明：

* limit\_low：段界限的低16位。
* base\_low：段基地址的低16位。
* base\_mid：段基地址的中间8位。
* access\_right：访问权限字段，定义了段的类型和访问权限。
* limit\_high：段界限的最高4位。
* base\_high：段基地址的最高8位。

## GATE\_DESCRIPTOR结构体

门描述符数据结构，用于定义中断或异常处理程序的入口点。



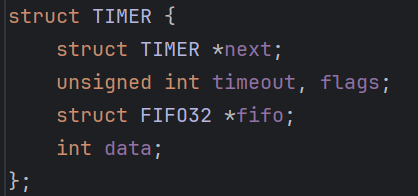
相关功能实现在dsctbl.c文件中。

成员变量说明：

* offset\_low：中断或异常处理程序的偏移地址的低16位。
* selector：段选择器，指向包含中断或异常处理程序代码的段。
* dw\_count：储存访问权限字段的高8位。
* access\_right：访问权限字段，定义了门的类型和访问权限。
* offset\_high：中断或异常处理程序的偏移地址的高16位。

## TIMER结构体

实现计时器对象的数据结构。



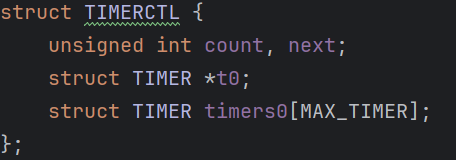
相关功能实现在dsctbl.c文件中。

成员变量说明：

* next：指向下一个TIMER结构体的指针，用于构建定时器链表。
* timeout：表示定时器的超时时间。
* flags：用于存储定时器的状态（0为未启用，1为启用）
* fifo：指向FIFO32先进先出队列，当定时器触发时，向该队列传输消息。
* data：用于存储与定时器触发时向FIFO32传输的数据。

## TIMERCTL结构体

计时器控制块，实现对计时器对象进行控制的数据结构。



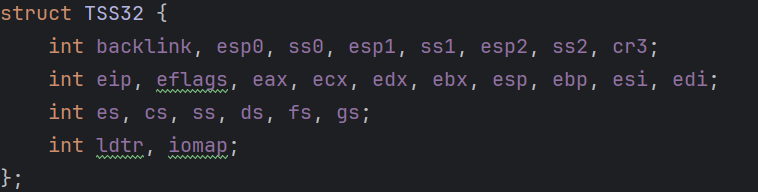
相关功能实现在dsctbl.c文件中。

成员变量说明：

* count：用于存储当前启用的定时器数量。
* next：表示下一个将要超时的定时器的时间。
* t0：指向下一个将要超时的计时器的指针，指向计时器链表的头部。
* timers0：TIMER结构体数组，存储系统中所有定时器对象。

## TSS32结构体

储存当前进程的处理器上下文的数据结构，包含了CPU在执行任务切换时需要保存和恢复的寄存器状态。



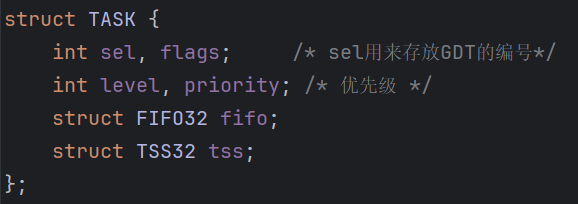
相关操作实现在mtask.c文件中

成员变量说明：

* backlink：指向先前任务的TSS的链接。
* esp0 到 ss2：分别为不同特权级别的堆栈指针和堆栈段选择器。
* cr3：存储当前任务的页目录基地址寄存器值。
* eip, eflags, eax 到 edi：通用寄存器和指令指针，保存当前任务的执行状态。
* es 到 gs：段寄存器。
* ldtr：局部描述符表寄存器。
* iomap：I/O位图的基地址，用于处理I/O权限。

## TASK结构体

用于表示进程控制块PCB的数据结构，包含了操作系统了解进程状态，以及对进程进行管理的所有必要信息。



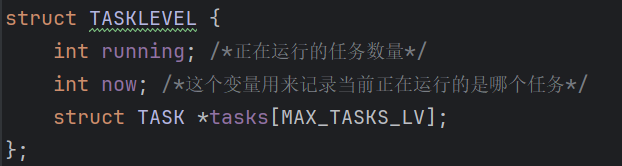
相关操作实现在mtask.c文件中

成员变量说明：

* sel：任务的全局描述符表（GDT）编号。
* flags：任务的标志位。
* level：任务的优先级级别。
* priority：任务的具体优先级。
* fifo：一个先进先出（FIFO）消息队列
* tss：任务的状态，通过TSS32结构体定义。

## TASKLEVEL结构体

某一特定优先级级别的就绪进程队列。



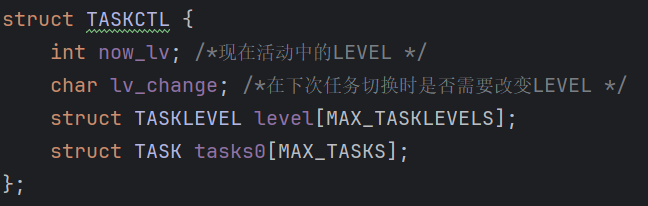
相关操作实现在mtask.c文件中

成员变量说明：

* running：当前级别上正在运行的任务数量。
* now：当前正在运行的任务的索引。
* tasks[MAX\_TASKS\_LV]：一个数组，存储指向当前级别上所有任务的指针。

## TASKCTL结构体

用于实现多级优先队列的数据结构，用于完成进程调度功能。



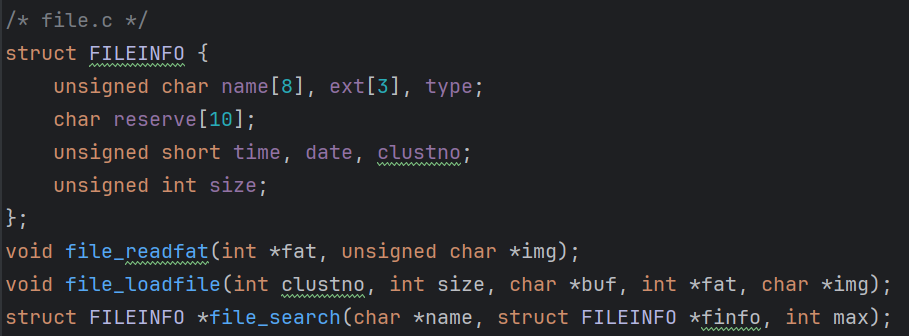
相关操作实现在mtask.c文件中

成员变量说明：

* now\_lv：当前活动的优先级级别。
* lv\_change：指示在下次任务切换时是否需要改变优先级级别。
* level[MAX\_TASKLEVELS]：每个元素是一个TASKLEVEL结构体，代表一个优先级级别。
* tasks0[MAX\_TASKS]：存储所有进程控制块的数组。

## FILEINFO结构体

用于表示文件系统中文件基本信息的数据结构。



相关操作实现在file.c文件中

成员变量说明：

* name[8]：用于存储文件的名称。
* ext[3]：用于存储文件的扩展名。
* type：用于存储文件的类型或属性。
* reserve[10]：10字节的保留字段
* time：表示文件的最后修改时间。
* date：表示文件的最后修改日期。
* clustno：表示文件在FAT12文件系统中的第一个簇的编号。
* size：表示文件的大小.

以上数据结构记录了内核程序所需要的关键信息，是内核程序中的重要变量。

# 运行测试

## 系统启动

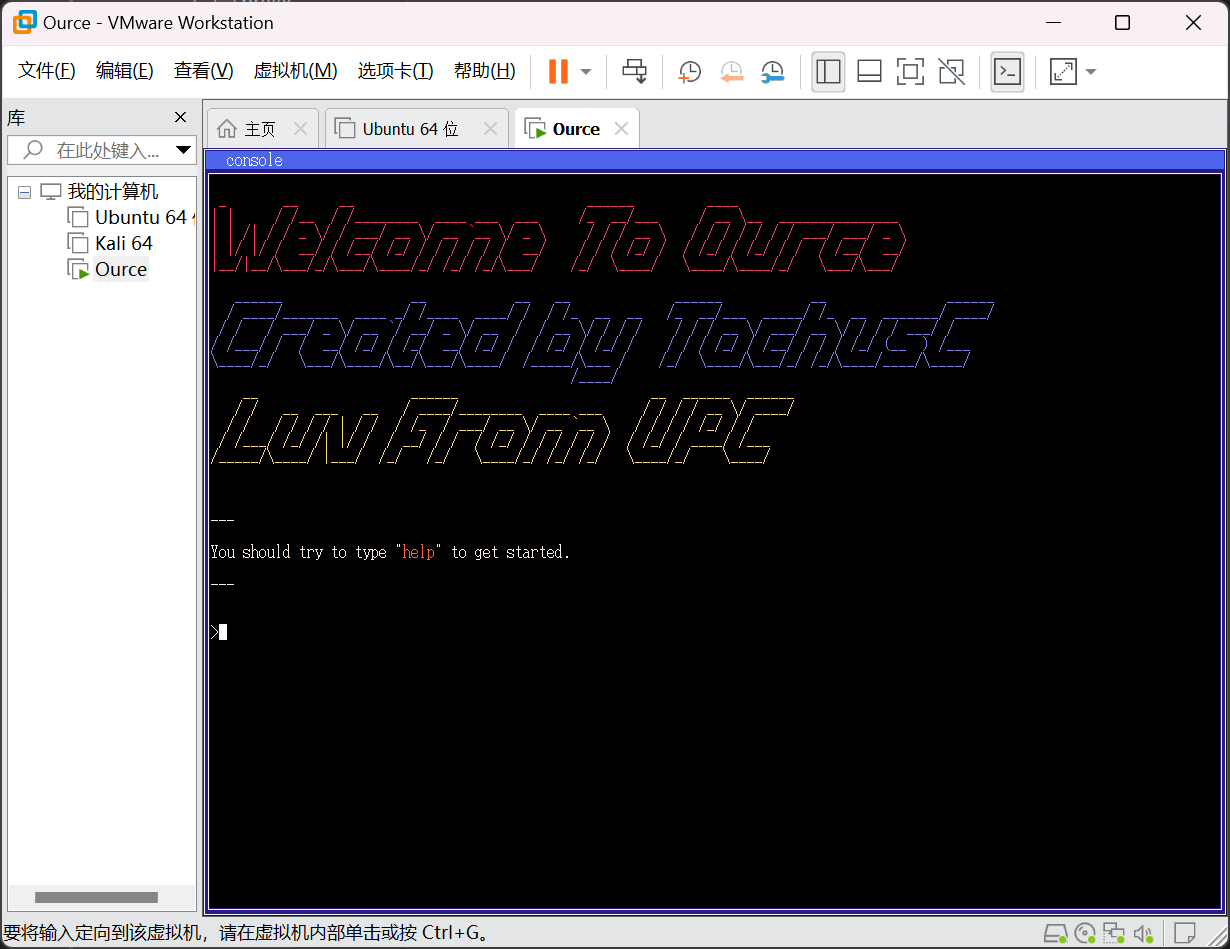


图 4 Ource启动画面

如图4所示，Ource成功启动并输出了欢迎信息及提示信息。

## 相关命令

### 输入help命令，查看帮助。

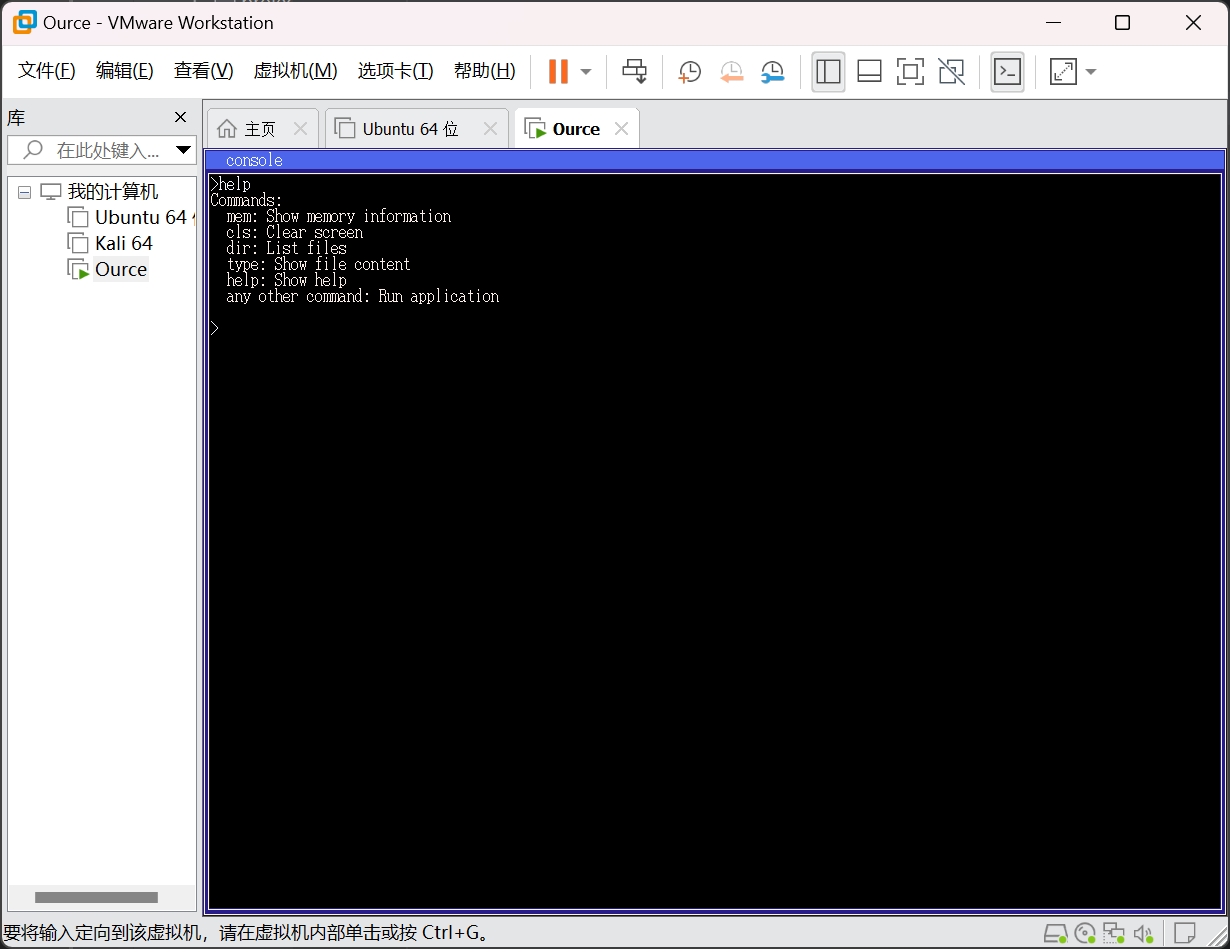


图 5 help命令运行结果

如图5所示，输入help命令后，命令行中列出了支持的所有指令。

### 输入mem命令，查看系统内存使用情况

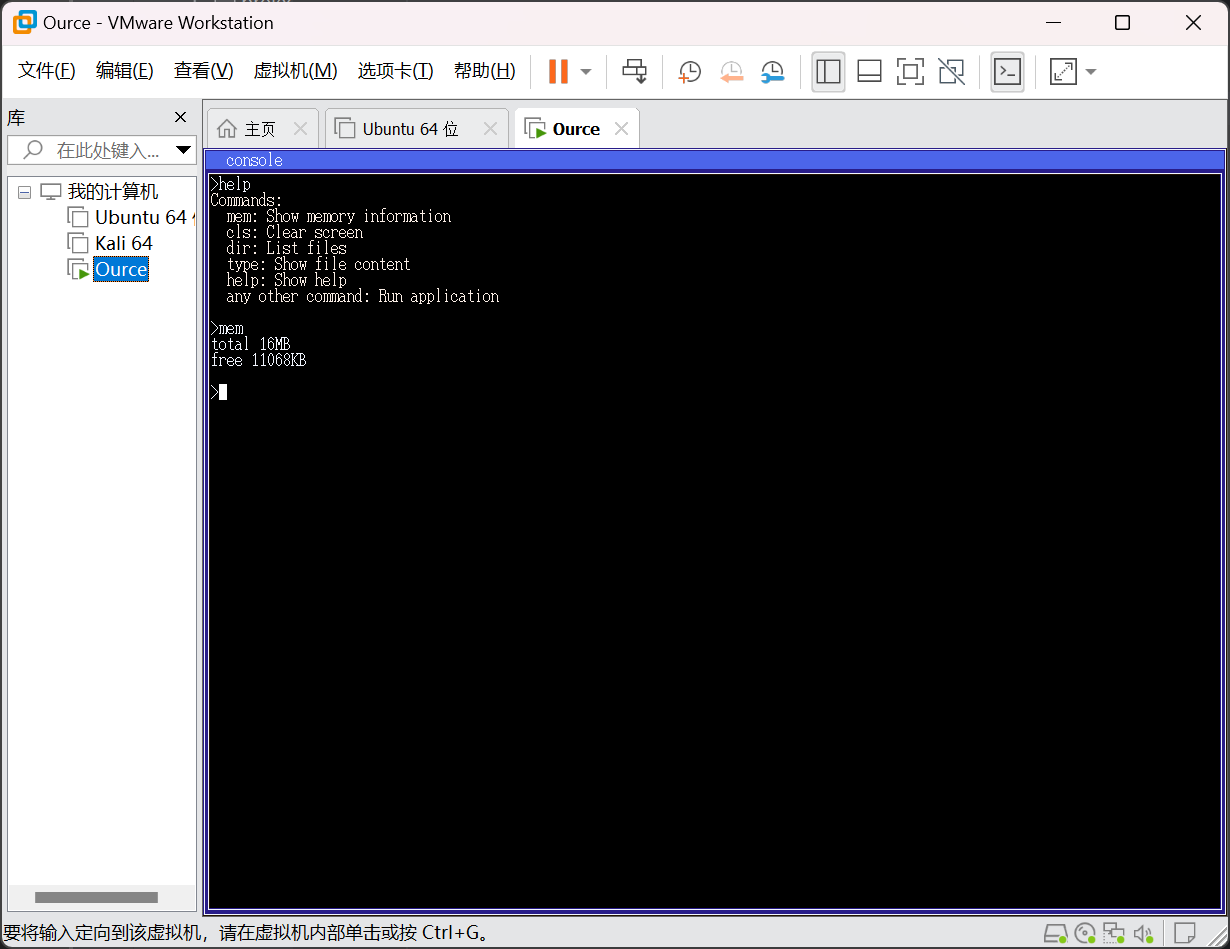


图 6 mem命令运行结果

如图6所示，mem命令输出了系统的所有内存和空闲内存情况。

### 输入dir命令，查看外存文件目录。

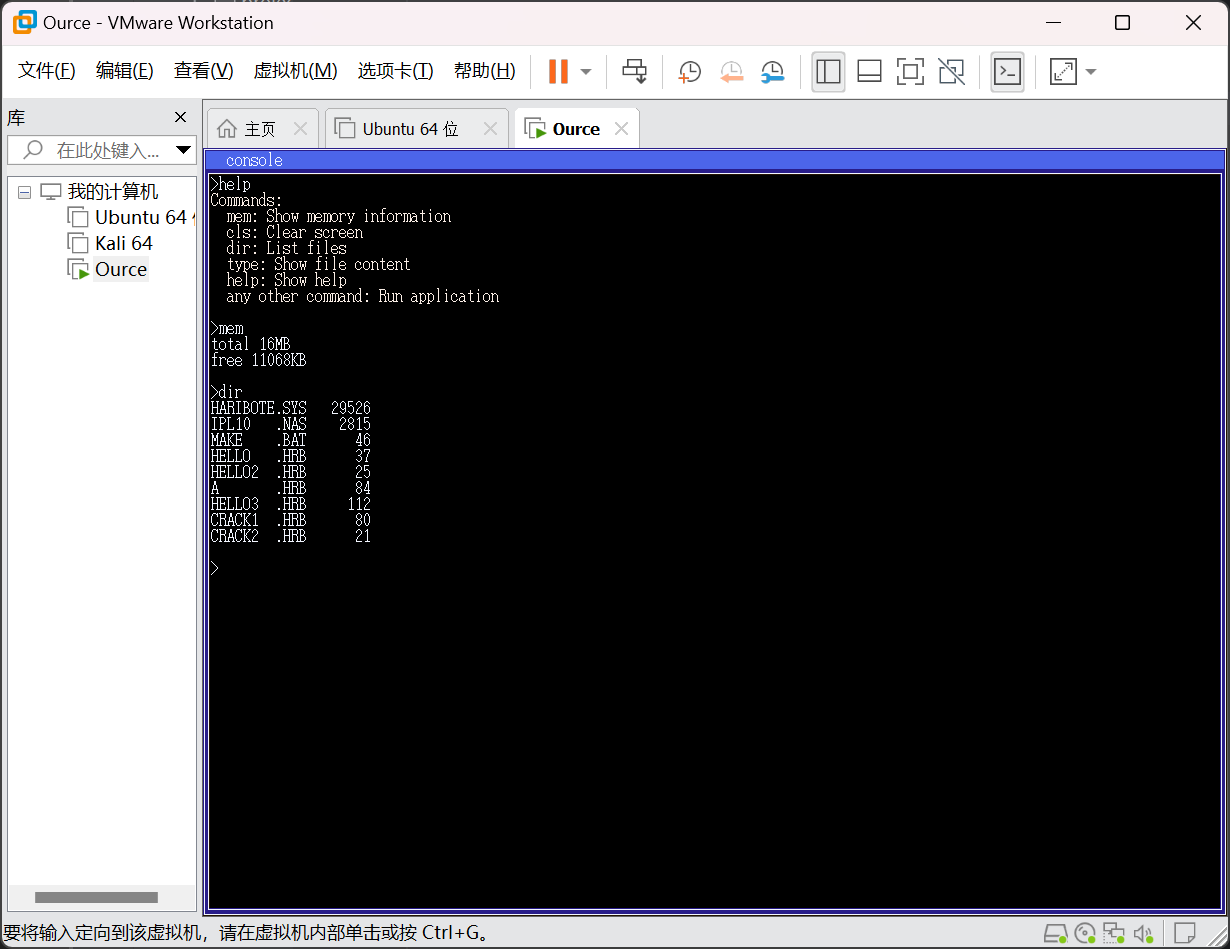


图 7 dir命令运行结果

从图7可以看到，dir命令列出了文件系统中所包含文件的基本信息。

### 通过type命令，查看某个文件的内容

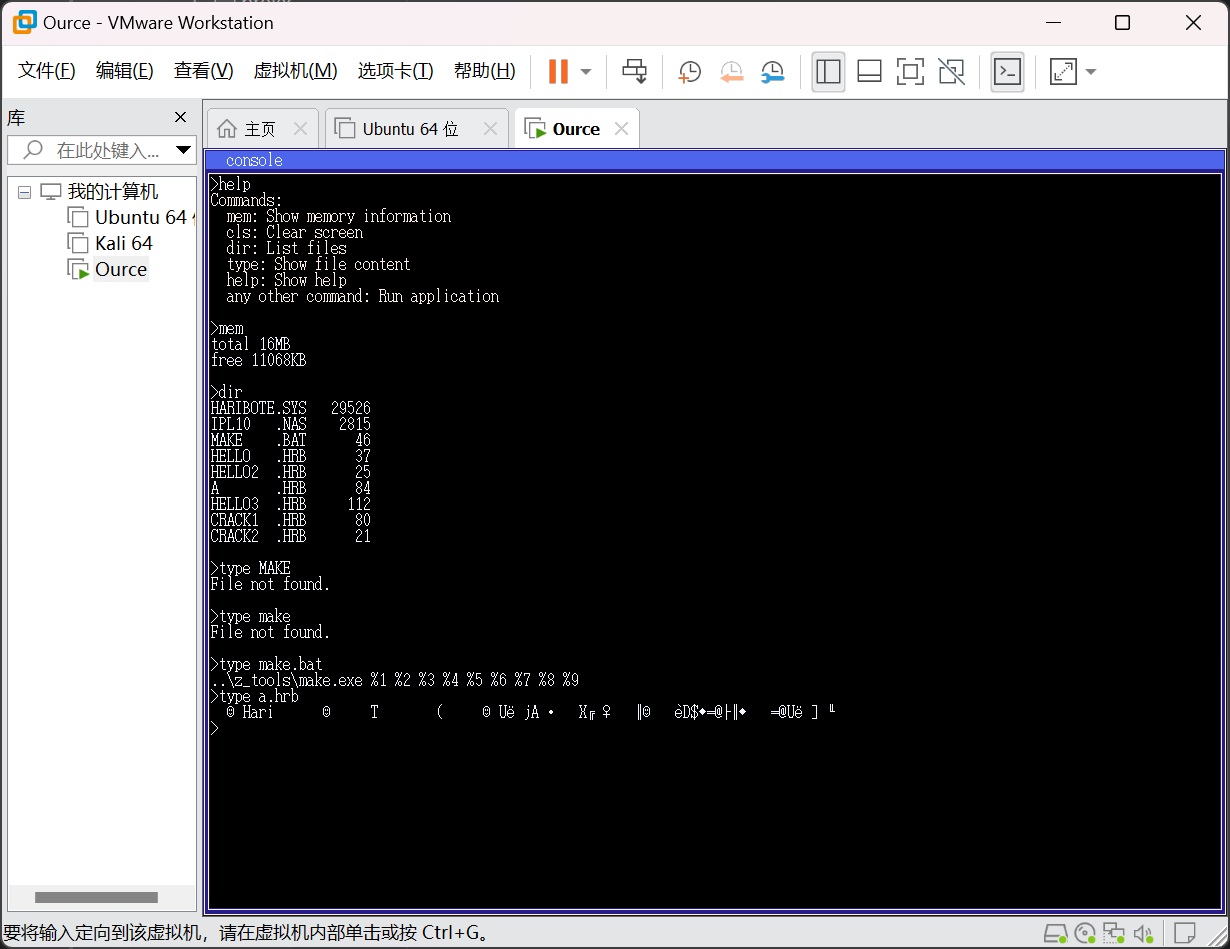


图 8 type命令运行结果

如图8所示，type命令输出了make.bat和a.hrb的内容，其中因为a.hrb为二进制格式的可执行文件，所以输出了乱码信息。

### 通过cls指令刷新屏幕

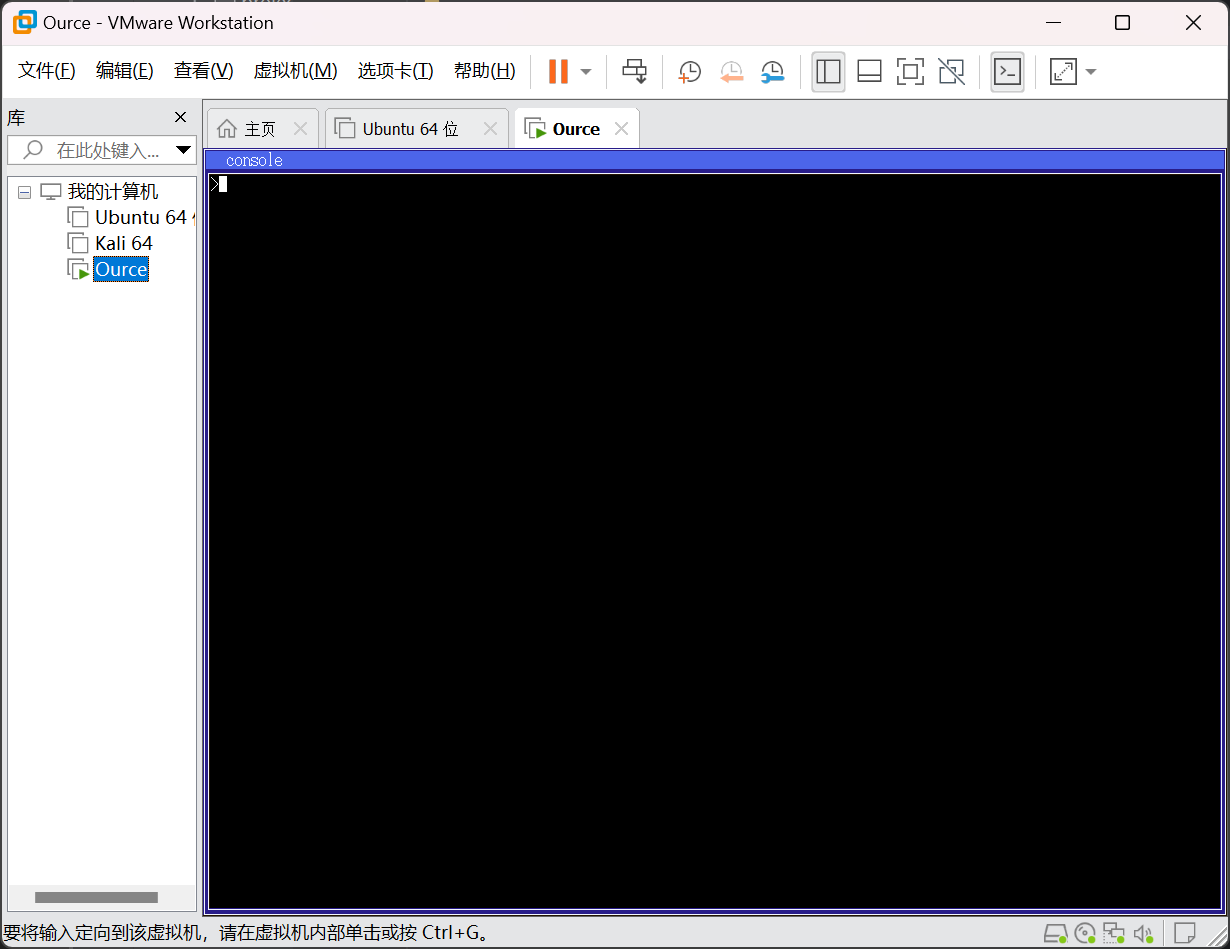


图 9 cls命令运行结果

如图9所示，cls指令刷新了命令行屏幕。

### 通过命令行运行.hrb可执行二进制文件

a.hrb是用C语言编写并编译的外部程序,其C源代码如下：

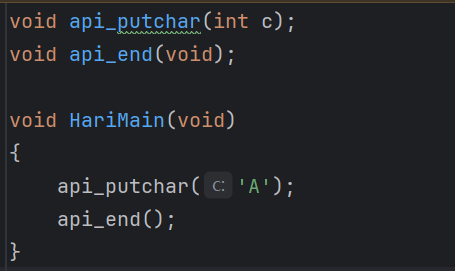


图 10 a.c内容

hello3.hrb是用汇编语言编写并汇编的外部程序，其汇编语言源代码如下：

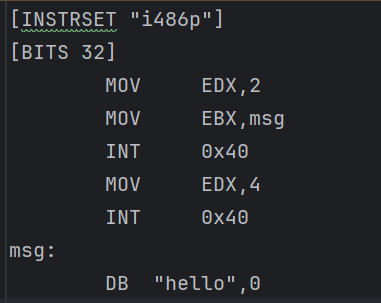


图 11 hello3.nas内容

通过Ource运行该两个外部程序，运行结果如图12所示。

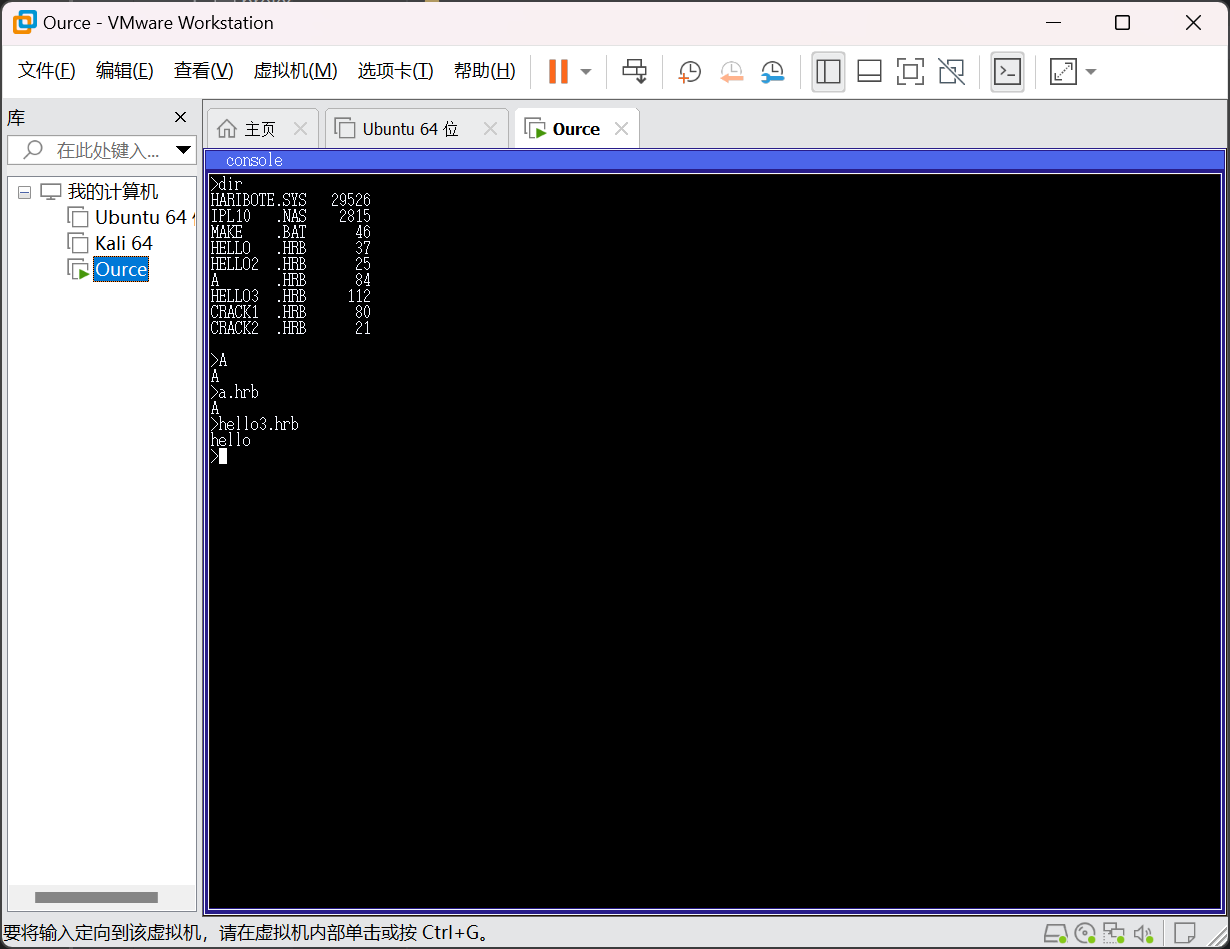


图 12 可执行二进制文件运行结果