即时开展 RISC-V Linux 内核与应用实验

贾献华<jiaxianhua@tinylab.org>
2022/12/10

泰晓科技 | TinyLab.org
RISC-V Linux 内核技术调研
https://tinylab.org/riscv-linux

目录

- 泰晓科技
  - Linux 内核观察 - v6.0
  - RISC-V Linux 内核剖析 - 2022
- Linux Lab Disk / Linux Lab 真盘
- RISC-V Linux 内核实验
- RISC-V Linux 应用实验

“RISC-V Linux 内核技术调研”开源项目

致力于 RISC-V + Linux 内核以及周边技术的学习、调研、总结与分享

开放资源

① 技术文章 https://tinylab.org/riscv-linux
② 协作仓库 https://gitee.com/tinylab/riscv-linux
③ 视频回放 https://space.bilibili.com/687228362
④ 技术专栏 https://zhihu.com/column/tinylab

实习兼职

① 加入指南 https://gitee.com/tinylab/riscv-linux
② 实验设备 https://tinylab.org/linux-lab-disk
③ 待领提案 https://gitee.com/tinylab/riscv-linux/issues
④ 联系我们 tinylab

当前成果（截止2022/11/05）

周期	35+
文章数	82+
在线分享/视频回放数	33+
上游项目贡献次数	10+
提案数	34+
实习生（累计）	7+
兼职工程师（累计）	11+

“泰晓科技”技术社区

技术原创社区，聚焦 Linux 十年

开放资源

① 社区网站 https://tinylab.org
② 代码仓库 https://gitee.com/tinylab 和 https://github.com/tinyclub
③ B 站视频 https://space.bilibili.com/687228362
④ 知乎专栏 https://www.zhihu.com/column/tinylab

付费服务

① 企业服务 https://tinylab.org/ruma.tech
② 视频课程 https://m.cctalk.com/inst/sh8qtdag
③ 知识星球 https://t.zsxq.com/uB2vJyF
④ 开源小店 https://shop155917374.taobao.com

联系方式


tinylab


泰晓科技


contact@tinylab.org

Linux Lab Disk / Linux Lab 真盘

- Linux Lab Disk / Linux Lab 真盘 - 泰晓科技 (tinylab.org)
- Linux Lab Disk 是由泰晓科技 Linux Lab 开源项目组研发的一种智能随身 Linux 系统盘，在原有随身 Linux 系统盘技术的基础上开创了多项特性，大大革新了用户使用体验。
- 零基础用户可以免安装在 1 分钟内即插即用用上 Linux 系统直接开展 Linux 内核与嵌入式 Linux 系统等实验，也可以当普通 Linux 系统使用，比如用来上网、做练习、开发、测试与比赛等等。
- Linux Lab Disk 又名 Linux Lab 真盘，除了独创的多项特性，还集成了自研的 Linux Lab, Linux 0.11 Lab 等实验环境。
- 首页-泰晓科技开源小店-淘宝网 (taobao.com)

产品特性

- 智能启动，开创了 3 种全新的智能化“傻瓜式”使用方式，可自动检测后并行启动、可免关机直接来回切换、还可以智能记忆自动启动。
  - 如何安装智能启动管理软件
- 相互套娃，多支 Linux Lab Disk 可相互启动或来回切换，因此，可根据喜好同时使用多个不同的 Linux 系统发行版。
- 透明倍容，透明提供接近翻倍的可用容量空间，“零成本”获得接近一倍的额外存储空间。
- 时区兼容，自动兼容 Windows, MacOS 和 Linux 的时区设定，跟主系统来回任意切换后时间保持一致。
- 自动共享，在 Windows 或 Linux 主系统下并行运行时，自动提供多种与主系统的文件与粘贴板共享方式。
- 零损编译，支持“半内存”与“全内存”的编译方式，可实现磁盘“零”写，极大地提升磁盘寿命，并提升实验效率。
- 即时实验，集成自研 Linux Lab，Linux 0.11 Lab 等实验环境，可在 1 分钟内开展 Linux 内核、嵌入式 Linux、U-Boot、汇编、C、Python、数据库、网络等实验。
- 出厂恢复，全系 6 大 Linux 发行版已全部支持出厂恢复功能，在“rm -rf /”后都能启动并恢复出厂系统，同时支持自动备份和急救模式，用起来更安心！

消除门槛：从入门到放弃

- 一个月:以往书籍的实验步骤
  - 安装步骤随着环境变化不兼容
  - 重新下载和安装繁琐又浪费时间
- 一分钟:安装到随身 U盘: Linux Lab 真盘
  - 免安装、免下载、可移动、可离线使用
  - 1分钟即可上手

提升效率:下载加速 + 高效编译、极速启动、轻松调试与测试

- 以往困境
  - 下载慢到不知道何时到头编译经常失败，缺这缺那 离看到实验效果的路径较远硬件调试需要特殊的工具，QEMU 调试不知道怎么用
- 下载部分
  - 精心配置或者提供了更快的国内资源镜像
- 编译部分
  - 集成了各种工具链，支持内存编译
  - 适配好了~100个左右内核版本(配置文件 + 内核 patch + 编译器配置)
    - 从0.11, 2.6到 6.0 ...
- 启动部分
  - 透明启动，参数分类预设好，更易调整
  - 预编译好了内核与文件系统，可直接复用
- 调试与测试
  - make debug

Linux Lab Disk 基础用法

- 列出并选择验证好的板子
  - $ make list
  - $ make BOARD=riscv64/virt
- 列出并选择验证好的内核版本 RISC-V
  - $ make list-kernel
  - v5.1 v5.13 v5.17 v5.17.9 v5.18.9 [v6.0.7]
  - $ make config LINUX=v5.1
  - or
  - $ make kernel-clone LINUX=V5.1 LINUX_NEW=v5.12
- 启动开发板
  - $ make boot ROOTDEV=/dev/nfs
- 开发 Linux
  - $ make kernel-menuconfig
  - $ make kernel
  - $ make boot

Linux Lab Disk 进阶用法

- 切换 gcc 编译器
  - $ make gcc-list
  - $ make gcc-switch CCORI=linaro
  - $ make gcc-switch CCORI=internal GCC=4.7
  - Or
  - $ make config CCORI=internal GCC=4.7
- 启用 llvm/clang
  - // Documentation/kbuild/1lmm.rst
  - $ make kernel LLVM=1 CLANG=1
- 共享文件:NFS或9pnet
  - $ make boot ROOTDEV=/dev/nfs
  - $ make boot SHARE=1
- 研究某个内核特性
  - $ make BOARD=x86_64/pc
  - $ make kernel-clone LINUX NEW=v5.13
  - $ make test f=rust m=rust print

Linux Lab Disk 进阶用法

- 编译: make kernel|root|uboot|qemu
  - 每个平台内建至少 1款交叉编译工具
  - 每个平台外置 3+ 款验证好的交叉编译工具
- 运行:make boot
  - 预编译 Bootloader、内核和文件系统镜像
  - 内建 QEMU 、 QEMU-User 和运行功能
- 调试:make debug
  - 内建基于 QEMU 的调试功能
- 测试:make test
  - 内建全方位自动化测试功能

Linux Lab Disk 进阶用法

- 调试内核
  - $ sudo tools/build/cache # 开启内存编译
  - $ make feature feature=debug
  - $ make kernel
  - $ make debug
- 测试内核
  - $ make test m=exception,hello TEST PREPARE=board-init,kernel-cleanup
  - $ make test m=lkdtm lkdtm_args='cpoint_name=DIRECT cpoint_type=EXCEPTION'
  - $ make DEBUG=1
  - $ make test TEST REBOOT=2
  - $ make test TEST TIMEOUT=50
  - $ make test feature=kft LINUX=v2.6.36 BOARD=malta TEST_PREPARE=prepare
  - $ make test TEST BEGIN=date TEST END=date TEST CASE=/tools/ftrace/trace.sh

RISC-V Linux 内核实验

- (实验一):快速上手
- (实验二):内核构建
- (实验三):开发内核模块
- (实验四):RealTime Linux

实验一:快速上手

- 进入实验环境后，确保当前工作路径为 /labs/linux-lab:
  - $pwd
  - /labs/linux-lab
- 输入如下命令，选择 riscv64/virt 虚拟开发板:
  - $make BOARD=riscv64/virt
- 输入如下命令，快速启动系统:
  - $ make boot

实验一:快速上手

- 2 行命令快速启动
  - make BOARD=riscv64/virt
  - make boot

```
Welcome to Linux Lab
linux-lab login: root
# uname -a
Linux linux-lab 6.0.7 #1 SMP Wed Nov 9 14:16:22 CST 2022 riscv64 GNU/Linux
#
```

- 只需要输入 root 然后输入回车即可：
  - Welcome to Linux Lab
  - linux-lab login: root
  - # uname -a
  - Linux linux-lab 6.0.7 #1 SMP Wed Nov 9 14:16:22 CST 2022 riscv64 GNU/Linux
  - # poweroff
- 部分开发板的关机功能不完善，可通过 Ctrl + a x
- (依次按下Ctrl 和A，同时释放，再单独按下x)来退出QEMU

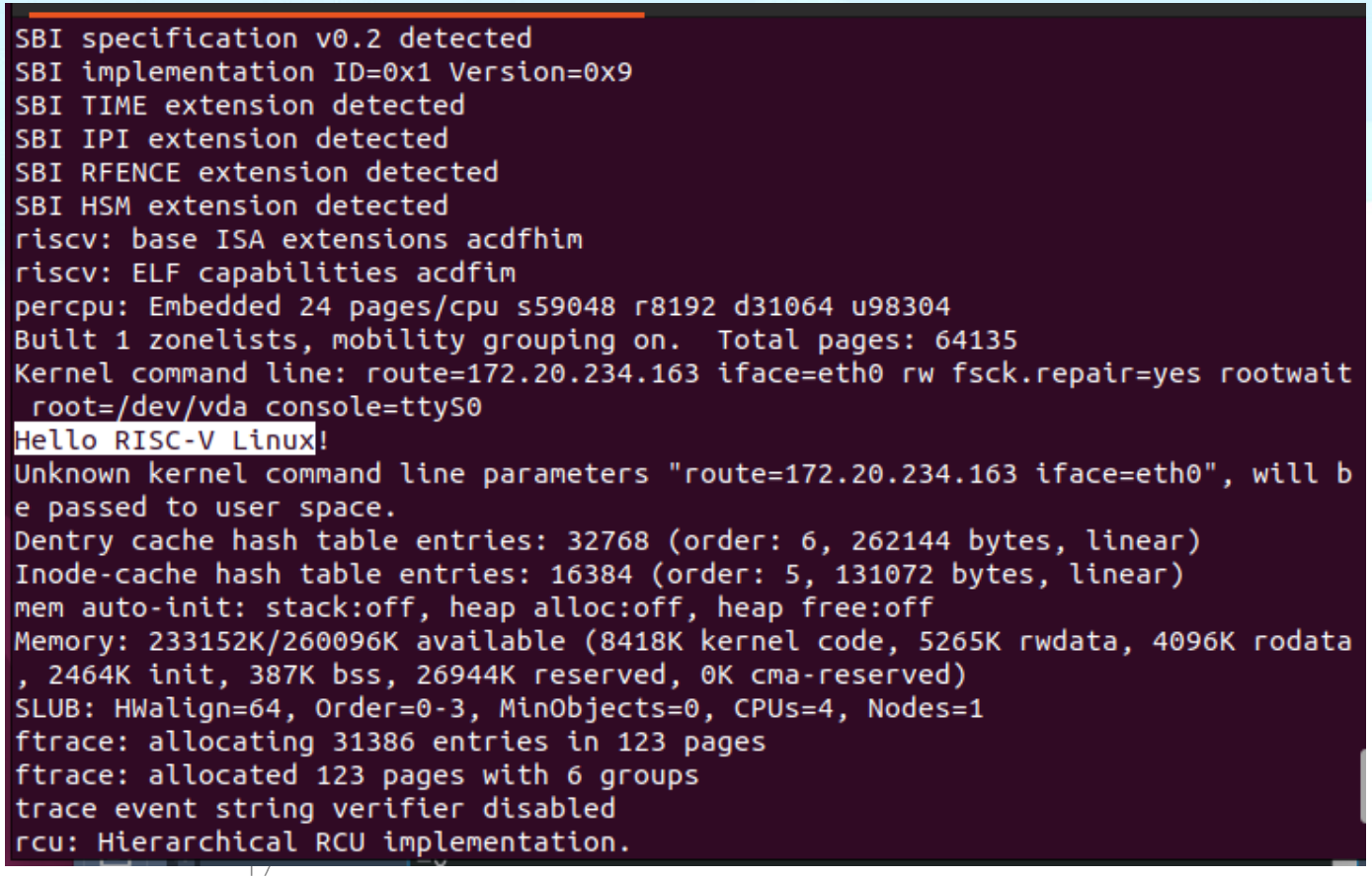
实验二: 内核构建

- 进入内核源码目录做一些修改，例如在 start_kernel() 中加点打印

- ubuntu@linux-lab:/labs/linux-lab/src/linux-stable$ git diff
- diff --git a/init/main.c b/init/main.c
- index 1fe7942f5d4a..4de4f32f4f5f 100644
- --- a/init/main.c
- +++ b/init/main.c
- @@ -961,6 +961,8 @@ asmlinkage __visible void __init __no_sanitize_address start_kernel(void)
- 	page_alloc_init();
-
- 	pr_notice("Kernel command line: %s\n", saved_command_line);
- +
- +	pr_info("Hello RISC-V Linux!\n");
- 	/* parameters may set static keys */
- 	jump_label_init();
- 	parse_early_param();

实验二：内核构建

- vi src/linux-stable/init/main.c # start_kernel()
- make kernel
- make boot



实验三:开发内核模块

- Linux Lab 下提供了多个内核模块例子，以 hello 模块为例
- 编译并安装该内核模块到根文件系统:
  - $ make modules M=src/modules/hello/
  - $ make modules-install M=src/modules/hello
- 启动新的根文件系统并通过 modprobe 来操作模块:
  - $ make boot ROOTDEV=/dev/nfs
  - $ ls /lib/modules/kernel-xxx/extra/hello.ko
  - $ modprobe hello
  - $ lsmod
  - $ modprobe -r hello

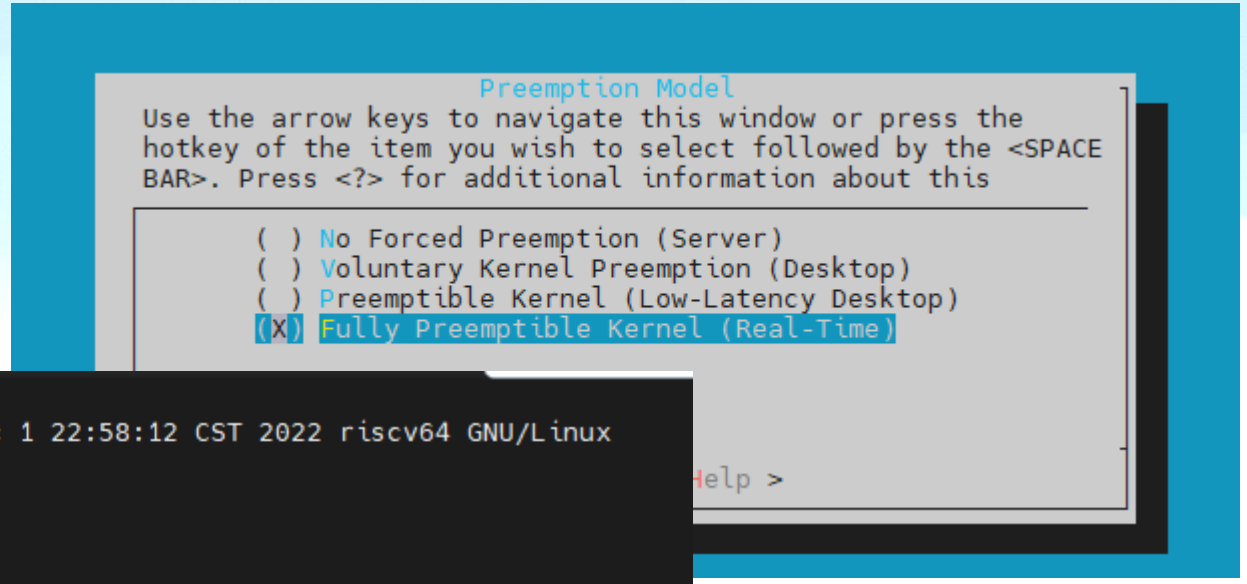
实验四:RealTime Linux

- articles/20221201-riscv-real-time-linux-1.md · 泰晓科技/RISCV-Linux - 码云 - 开源中国 (gitee.com)
- 源代码仓库是干净的
  - $ make kernel-cleanup
  - $ make kernel-cleanall
- 打上 RT Patches
  - $ mv patch-6.0-rt11.patch 0.patch-6.0-rt11.patch
  - $ mv v2_20220901_jszhang_riscv_add_preempt_rt_support.mbx 1.v2_20220901_jszhang_riscv_add_preempt_rt_support.mbxnall
  - $ make kernel-patch

实验四:RealTime Linux

Linux linux-lab 6.0.7-rt11-dirty #1 SMP PREEMPT_RT Thu Dec 1 22:58:12 CST 2022 riscv64 GNU/Linux

- 配置使能 PREEMPT_RT
  - make linux-menuconfig
  - /PREEMPT_RT




```
# uname -a
Linux linux-lab 6.0.7-rt11-dirty #1 SMP PREEMPT_RT Thu Dec 1 22:58:12 CST 2022 riscv64 GNU/Linux
# zcat /proc/config.gz | grep PREEMPT
CONFIG_HAVE_PREEMPT_LAZY=y
CONFIG_PREEMPT_LAZY=y
# CONFIG_PREEMPT_NONE is not set
# CONFIG_PREEMPT_VOLUNTARY is not set
# CONFIG_PREEMPT is not set
CONFIG_PREEMPT_RT=y
CONFIG_PREEMPT_COUNT=y
CONFIG_PREEMPTION=y
CONFIG_PREEMPT_RCU=y
CONFIG_DEBUG_PREEMPT=y
CONFIG_PREEMPTIRQ_TRACEPOINTS=y
CONFIG_TRACE_PREEMPT_TOGGLE=y
CONFIG_PREEMPT_TRACER=y
# CONFIG_PREEMPTIRQ_DELAY_TEST is not set
#
```

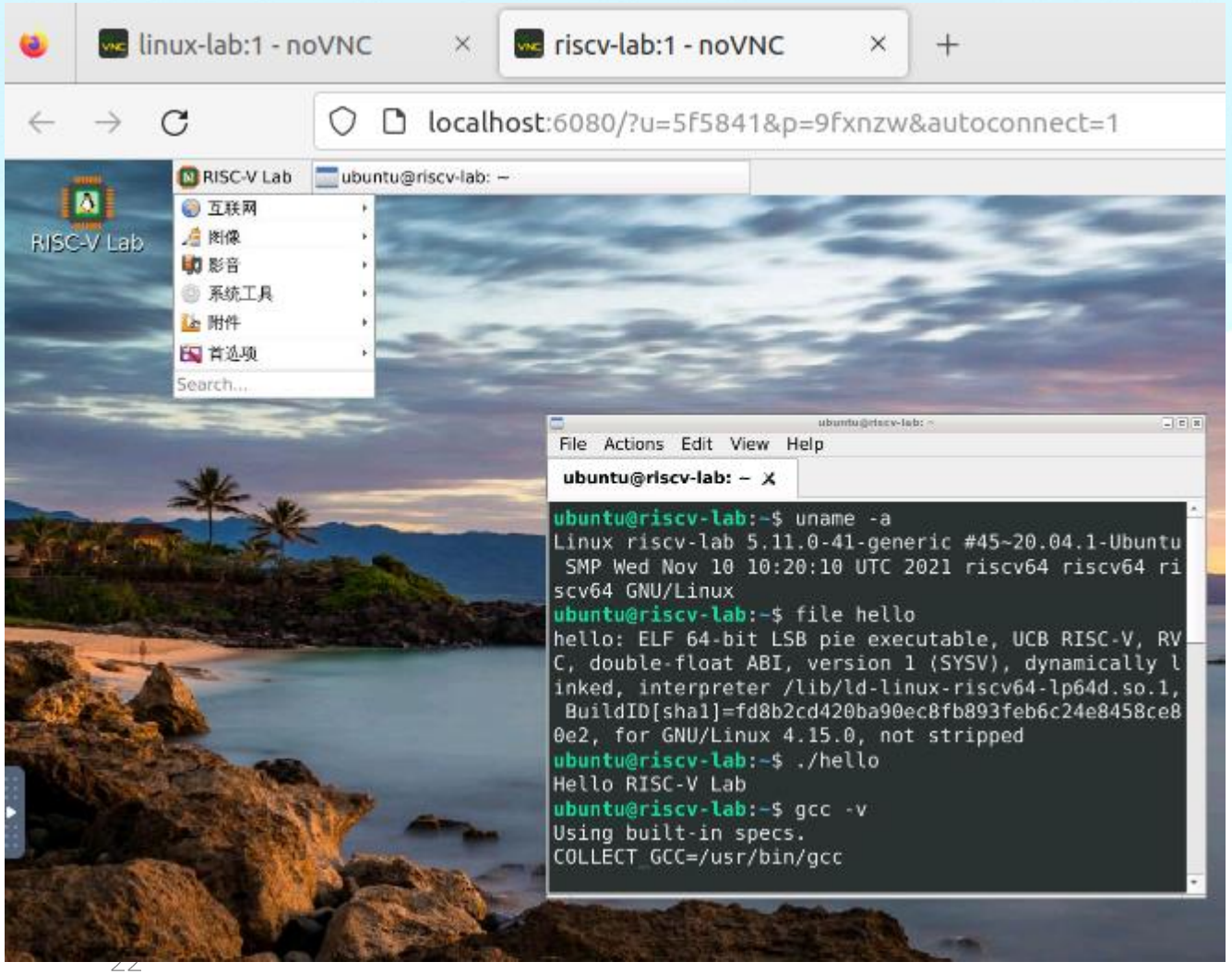
RISC-V Linux 应用实验

- RISC-V Lab GUI
- RVOS Lab
- Bash/Assembly/C/Makefile
  - toolchain

RISC-V Lab GUI

```
// 启用 lxqt 桌面
$ DESKTOP=lxqt tools/docker/run riscv-lab

// 用回 xfce 桌面
$ DESKTOP=xfce tools/docker/rerun riscv-lab
```



RVOS Lab

- rvos-lab: Support of https://gitee.com/unicornx/riscv-operating-system-mooc for Linux Lab.
- 批量运行实验：
  - $ git clone https://gitee.com/tinylab/rvos-lab
  - $ cd rvos-lab/code/os
  - $ make run

RVOS Lab

- 批量运行截图

```
01-helloRVOS     04-multitask      07-hwtimer     10-swtimer  Makefile
02-memanagement  05-traps          08-preemptive  11-syscall
ubuntu@linux-lab:/labs/linux-lab/src/examples/rvos-lab/code/os$ make run
begin run ALL exercises for os samples .....................
for dir in 00-bootstrap 01-helloRVOS 02-memanagement 03-contextswitch 04-multitask 05-traps 06-interrupt
s 07-hwtimer 08-preemptive 09-lock 10-swtimer 11-syscall ; do make -C $dir run; done
make[1]: Entering directory '/labs/linux-lab/src/examples/rvos-lab/code/os/00-bootstrap'
Press Ctrl-A and then X to exit QEMU
---------------------------------
QEMU: Terminated
make[1]: Leaving directory '/labs/linux-lab/src/examples/rvos-lab/code/os/00-bootstrap'
make[1]: Entering directory '/labs/linux-lab/src/examples/rvos-lab/code/os/01-helloRVOS'
Press Ctrl-A and then X to exit QEMU
---------------------------------
Hello, RVOS!
QEMU: Terminated
make[1]: Leaving directory '/labs/linux-lab/src/examples/rvos-lab/code/os/01-helloRVOS'
make[1]: Entering directory '/labs/linux-lab/src/examples/rvos-lab/code/os/02-memanagement'
Press Ctrl-A and then X to exit QEMU
---------------------------------
Hello, RVOS!
HEAP_START = 80003480, HEAP_SIZE = 07ffcb80, num of pages = 32756
TEXT:   0x80000000 -> 0x80002d20
RODATA: 0x80002d20 -> 0x80002e87
DATA:   0x80003000 -> 0x80003000
BSS:    0x8000308c -> 0x80003480
HEAP:   0x8000c000 -> 0x88000000
QEMU: Terminated
make[1]: Leaving directory '/labs/linux-lab/src/examples/rvos-lab/code/os/02-memanagement'
make[1]: Entering directory '/labs/linux-lab/src/examples/rvos-lab/code/os/03-contextswitch'
Press Ctrl-A and then X to exit QEMU
---------------------------------
Hello, RVOS!
HEAP_START = 800048fc, HEAP_SIZE = 07ffb704, num of pages = 32755
TEXT:   0x80000000 -> 0x80002f84
RODATA: 0x80002f84 -> 0x80003128
DATA:   0x80004000 -> 0x80004000
BSS:    0x8000408c -> 0x800048fc
HEAP:   0x8000d000 -> 0x88000000
Task 0: Created!
Task 0: Running...
Task 0: Running...
Task 0: Running...
Task 0: Running...
Task 0: Running...
Task 0: Running...
Task 0: Running...
Task 0: Running...
Task 0: Running...
```

更多实验

Linux Lab v1.0 中文手册

- README.md
- assembly
- gui-lab
- linux-0.11-lab
- python
- riscv-linux

src/examples

- shell
- c
- guilab
- makefile
- rvos-lab
- x86os-lab

致谢

赞助单位

ISCAS 中国科学院软件研究所
Institute of Software Chinese Academy of Sciences

Linux Lab Disk

该二维码 7 天内 (12月17日前) 有效，重新进入将更新