uCore for LoongArch32 Tutorial Book

None

None

None

Table of contents

1. uCore OS for LoongArch32

-
-
-
-
-
-
-
-
-
- Chiplab

2. 0

2.1

2.1.1

- LoongArch32
- (Docker)
-
-
- C
- LoongArch32

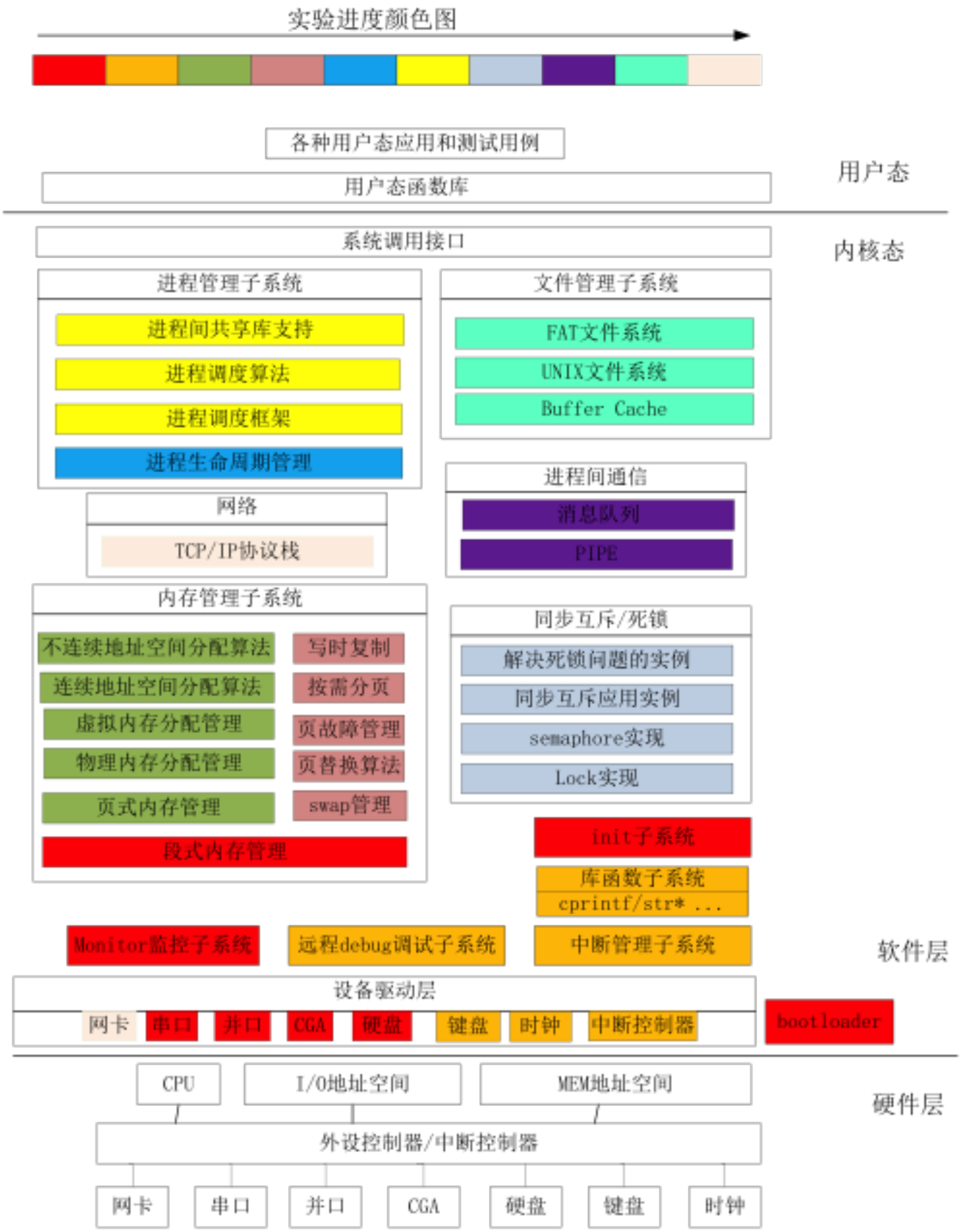
2.1.2 OS

Linux Windows Thompson UNIX Linus 21 Linux “ ”

MIT Frans Kaashoek 2006 PDP-11 UNIX Version 6 X86 xv6 MIT License xv6 “ ” “ ” “ ” —ucore “ ” C+asm 5K “ ”

ucore LoongArch32 LoongArch32 QEMU ucore GCC gcc gas ld MAKE Visual Studio Code Git SVN Meld deubg gdb qemu Docker Docker

ucore 1. TLB 2. LoongArch32 3. TLB 4. 5. 6. 7. 8. IO VFS buffer cache disk driver



1 ucore

2.1.3 OS lab

1. Makefile `LAB CONFIG`
2. `make clean`
3. uCore uCore
4. `make qemu -j 16` 16 CPU
5. `make debug` `make gdb` gdb

> ⚠ **Warning**
>
> Lab0

2.2

2.2.1

“ ”

2.2.2 LoongArch32

LoongArch 32 64 32

LoongArch32 32 r0~r31 ABI

`$r0`	`$zero`	`0`
`$r1`	`$ra`	
`$r2`	`$tp`	
`$r3`	`$sp`	
`$r4` - `$r5`	`$a0` - `$a1`	
`$r6` - `$r11`	`$a2` - `$a7`	
`$r12` - `$r20`	`$t0` - `$t8`	
`$r21`		
`$r22`	`$fp` / `$s9`	/
`$r23` - `$r31`	`$s0` - `$s8`	

LoongArch32 ABI `v0` `v1` `a0` `a1` `$r4` `$r5` ABI

32 _v1.02.pdf

marco

marco ISA marco

uCore for LoongArch32 marco `li.w` LoongArch32 32 32 `li.w` `lu12i.w` `ori` li.w

marco https://gitee.com/loongson-edu/la32r_binutils/blob/master/opcodes/loongarch-opc.c

opcodes RISC-V

LoongArch32

MIPS

x86 PIO

RISC-V MMU SBI Supervisor

LoongArch32 x86 ARM RISC-V

TLB MIPS TLB TLB 0 TLB

x86 ARM RISC-V CSR x86 CR3 RISC-V satp TLB

LoongArch32 MIPS TLB

x86 ARM RISC-V **TLB**

2.2.3

CPU x86-64 RAX RSP RBP ARM/MIPS/RISC-V 32 a0 a1 sp ra

CPU Control/Status Registers CSR CSR

LoongArch32 PLV0 PLV3 x86 Ring0 Ring3 ARM EL0 EL3 RISC-V Machine User

Note

RISC-V Supervisor

CSR `CRMD` `DMW0` `DMW1` 32 _v1.02.pdf

Info

32 _v1.02.pdf P55, CRMD

DMW DMW

2.2.4 I/O

I/O

I/O

I/O 3

PMIO (PORT-MAPPED I/O)

I/O LoongArch32 RISC-V ARM MIPS

x86 x86-64 I/O `in[blw]` `out[blw]` I/O GPR

MMIO (MEMORY-MAPPED I/O)

I/O I/O I/O

MMIO MMIO IO

UART IO

DMA (DIRECT MEMORY ACCESS)

MMIO CPU

CPU

IO GPU DMA MMIO

DMA IO IO DMA DMA

Note

1. CPU Cache MMIO Cache
2. CPU MMIO
3. MMIO CPU
4. CPU Cache DMA Cache Cache DMA CPU DMA CPU
5. C MMIO **volatile** MMIO O2
6. DMA Uncached Cache SIMD

E820 ACPI

x86 BIOS e820 x86 int BIOS

x86 Linux dmesg BIOS e820 GB reserved e820 BIOS e820

ACPI Advanced Configuration and Power Interface 90 x86 ACPI

ACPI Bootloader Linux `arch/*/boot/dts`

USB PCI-E

LoongArch32 uCore ACPI Bootloader

LoongArch32 0 32M

IO uCore MMIO QEMU Chiplab FPGA SoC

2.2.5

LoongArch32 DMW TLB DMW TLB TLB REFILL

DMW

LoongArch32 DMW `[31:29]` `[31:29]` Cache

32 _v1.02.pdf P43

TLB

TLB Cache

TLB Page Table Cache TLB

x86 ARM RISC-V TLB LoongArch32 MIPS TLB 32 _v1.02.pdf P55

TLB

DMW

MIPS Bootloader Bootloader LoongArch32 QEMU uCore DMW CSR.CRMD Chiplab Bootloader PMON PMON DMW

uCore DMW DRAM IO

DMW	MAT	PLV0	PLV3	PSEG	VSEG	VSEG
DMW0	1	1	0	0b000 (0x00000000-0x1fffffff)	0b101	`0xa0000000-0xbfffffff`
DMW1	0	1	0	0b000 (0x00000000-0x1fffffff)	0b100	`0x80000000-0x9fffffff`

QEMU DMW Chiplab FPGA PMON PMON DMW DMW PMON

LoongArch32 QEMU 3 DMW LoongArch32 Chiplab uCore TLBREFILL QEMU

2.2.6 QEMU

QEMU LoongArch32 QEMU

2.3

MIT xv6 Harvard OS161 Linux ucore OS OS Linux

2.3.1

x86-64 Docker

LoongArch32 QEMU Docker uCore Docker Hub.

Windows WSL2 Linux Docker Linux macOS Apple Silicon Linux Docker

Linux Docker

Note

WSL2+Ubuntu 22.04

Docker Daemon iptables-legacy

```
sudo apt install iptables
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
sudo service docker start
```

WSL1 Docker WSL2

x86-64

x86-64 ARM Apple Silicon Mac Linux ARM Debian Ubuntu qemu-user x86-64 Linux x86

```
sudo apt install qemu-user qemu-user-static binfmt-support gcc-x86-64-linux-gnu binutils-x86-64-linux-gnu binutils-x86-64-linux-gnu-dbg build-essential
```

x86-64 Linux

2.3.2 Docker

1. Docker Hub Docker Hub Docker Hub
2. Docker

Linux root sudo Docker VSCode Docker Docker

```
sudo usermod -aG docker $USER
newgrp docker
```

`newgrp docker`

2.3.3 Docker

1. `chenyy/la32r-env`

```
docker pull chenyy/la32r-env
```

2. `chenyy/la32r-env` `la32r-docker`

```
docker run -dit \
    --name la32r-docker \
    --user=$(id -u $USER):$(id -g $USER) \
    --net=host \
    --workdir="/home/$USER" \
    --volume="/home:/home" \
    --volume="/root:/root" \
    --volume="/mnt:/mnt" \
    --volume="/etc/group:/etc/group:ro" \
    --volume="/etc/passwd:/etc/passwd:ro" \
    --volume="/etc/shadow:/etc/shadow:ro" \
    --volume="/etc/sudoers.d:/etc/sudoers.d:ro" \
    -e LANG=en_US.UTF-8 \
    -e LANGUAGE=en_US.UTF-8 \
    -e LC_ALL=en_US.UTF-8 \
    chenyy/la32r-env
```

Docker /home Docker Docker Docker

3. `la32r-docker`

```
docker start la32r-docker
```

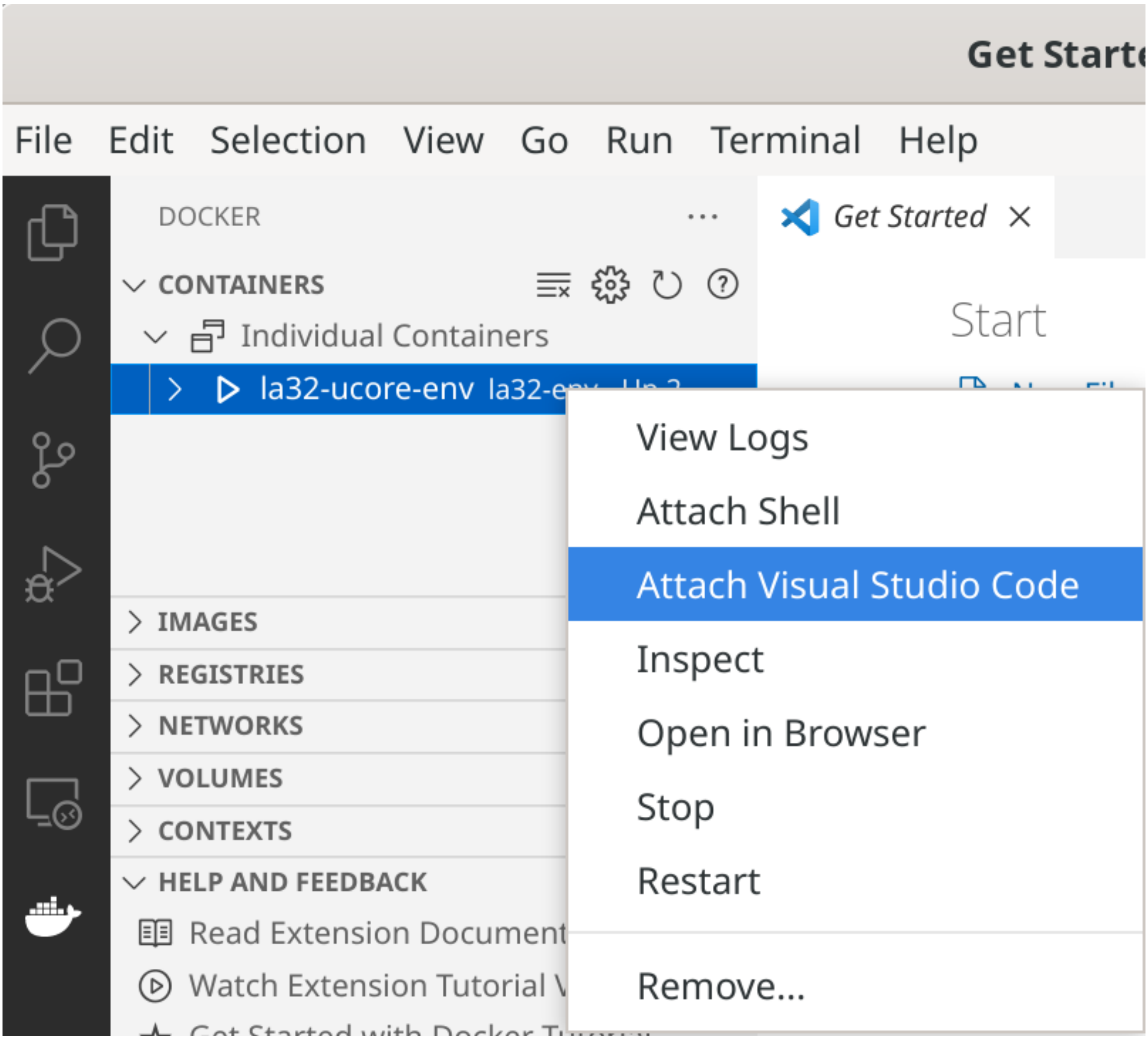
4. `la32r-docker` Shell

```
docker exec -it la32r-docker /bin/zsh
```

2.3.4 VSCode

VSCode Docker

VSCode Docker Attach Shell Attch VSCode Host VSCode Docker



2.3.5

Visual Studio Code Docker LoongArch32

git

Git

2.4

2.4.1

	Linux NOMMU
	/
BSS	

ABI Application Binary Interface

C GPR sp C

C

```c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

```
➜  gcc hello.c -c -o hello.o
➜  objdump -h hello.o

hello.o:     file format elf64-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         0000001a  0000000000000000  0000000000000000  00000040  2**0
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  0000005a  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  0000005a  2**0
                  ALLOC
  3 .rodata       0000000c  0000000000000000  0000000000000000  0000005a  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      0000001f  0000000000000000  0000000000000000  00000066  2**0
                  CONTENTS, READONLY
  5 .note.GNU-stack 00000000  0000000000000000  0000000000000000  00000085  2**0
                  CONTENTS, READONLY
  6 .eh_frame     00000038  0000000000000000  0000000000000000  00000088  2**3
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, DATA
```

`.text` `.data` `.bss` `.rodata`

- `.text`
- `.data` `static`
- `.bss` ELF
- `.rodata` const

ELF

“ ” C Makefile

ELF

ld

2.4.2

```
.extern main
.text
.globl _start
_start:
    # Config direct window and set PG
    li.w    $t0, 0xa0000011
    csrwr   $t0, 0x180
    /* CSR_DMWIN0(0x180): 0xa0000000-0xbfffffff->0x00000000-0x1fffffff Cached */
    li.w    $t0, 0x80000001
    /* CSR_DMWIN1(0x181): 0x80000000-0x9fffffff->0x00000000-0x1fffffff Uncached */
    # Enable PG
    li.w    $t0, 0xb0
    csrwr   $t0, 0x0
    /* CSR_CRMD(0x0): PLV=0, IE=0, PG */
    la  $sp, bootstacktop
    la  $t0, main
    jr  $t0
poweroff:
    b poweroff
_stack:
.section .data
    .global bootstack
bootstack:
    .space 1024
    .global bootstacktop
bootstacktop:
    .space 64
```

`start.S`

CSR DMWIN CSR_CRMD main main extern main

C

LoongArch32 QEMU ns16550a 0x1fe001e0

MMIO ns16550a

```c
#define UART_BASE 0x9fe001e0
#define UART_RX     0   /* In:  Receive buffer */
#define UART_TX     0   /* Out: Transmit buffer */
#define UART_LSR    5   /* In:  Line Status Register */
#define UART_LSR_TEMT       0x40 /* Transmitter empty */
#define UART_LSR_THRE       0x20 /* Transmit-hold-register empty */
#define UART_LSR_DR         0x01 /* Receiver data ready */

void uart_put_c(char c) {
    while (!(*((volatile char*)UART_BASE + UART_LSR) & (UART_LSR_THRE)));
    *((volatile char*)UART_BASE + UART_TX) = c;
}

void print_s(const char *c) {
    while (*c) {
        uart_put_c(*c);
        c ++;
    }
}

void main() {
    print_s("\nHere is my first bare-metal machine program on LoongArch32!\n\n");
}
```

`main.c`

0x1fe001e0 0x9fe001e0 DMW 0x80000000-0x9fffffff Uncached

Warning

Cached DMWIN0 0xbfe001e0 QEMU Cache CPU

```
SECTIONS
{
    . = 0xa0000000;
    .text : { *(.text) }
    .rodata : { *(.rodata) }
    .bss : { *(.bss) }
}
```

`lab0.ld`

0xa0000000 DMWIN0 Cache

Makefile

```makefile
TOOL    :=  loongarch32r-linux-gnusf-
CC      :=  $(TOOL)gcc
OBJCOPY :=  $(TOOL)objcopy
OBJDUMP :=  $(TOOL)objdump
QEMU    :=  qemu-system-loongarch32

.PHONY: clean qemu

start.elf: start.S main.c lab0.ld
	$(CC) -nostdlib -T lab0.ld start.S main.c -O3 -o $@

qemu: start.elf
	$(QEMU) -M ls3a5k32 -m 32M -kernel start.elf -nographic

clean:
	rm start.elf
```

Warning

Makefile "\t" Make "missing separator."

`Makefile`

Makefile Makefile `-nostdlib` stdlib

`start.S` `main.c` `lab0.ld` `Makefile` `make qemu`

```
➜  make qemu
qemu-system-loongarch32 -M ls3a5k32 -m 32M -kernel start.elf -nographic
loongson32_init: num_nodes 1
loongson32_init: node 0 mem 0x2000000

Here is my first bare-metal machine program on LoongArch32!
```

"Here is my first bare-metal machine program on LoongArch32!"

QEMU

nographic ++ctrl+a++ ++x++

Note

1. start.S
2. QEMU CPU

2.5

2.5.1

OS

Makefile

1. GCC `-O3`
2. GCC `-g`

Makefile

```
10c10
<       $(CC) -nostdlib -T lab0.ld start.S main.c -g -o $@
---
>       $(CC) -nostdlib -T lab0.ld start.S main.c -O3 -o $@
```

`make clean` `make`

QEMU GDB

GDB attach gdb QEMU GDB QEMU GDB GDB Remote Serial Protocol QEMU TCP Socket GDB

QEMU `-s` `-gdb tcp::1234` GDB Remote Serial Protocol TCP 1234

QEMU GDB QEMU `-S` QEMU

QEMU

- `-S` CPU
- `-s` -gdb tcp::1234 gdbserver 1234

ucore QEMU

```
qemu-system-loongarch32 -M ls3a5k32 -m 32m -kernel start.elf -nographic -S -s
```

GDB

gdb VSCode Terminal screen/tmux

lab0 `gdb` `

```
loongarch32r-linux-gnusf-gdb start.elf
```

QEMU

```
target remote 127.0.0.1:1234
```

```
tar rem :1234
```

GDB QEMU

GDB

- layout [src/asm/split]

  gdb src asm split

- breakpoint [target]

  `b`

  target / :

  `uart_put_c`

  ```
  b uart_put_c
  ```

  main.c 10

  ```
  b main.c:10
  ```

  PC `0xa0000004`

  ```
  b *0xa0000004
  ```

- info break

- info registers

- continue

  `c`

  QEMU `c` QEMU

- backtrace

  `bt`

- next

  `n`

- step

  `s`

- stepi

  `si`

- print [expr]

  `p`

  [expr] gdb

  `uart_put_c` `char c`

  ```
  p c
  ```

  c

  ```
  p/x c
  ```

  `0xa0000129` char

  ```
  p *(char*)0xa0000129
  ```

- display [expr]

  `disp`

  print print disp gdb

  gdb

.gdbinit

gdb target remote 127.0.0.1:1234 layout

gdbinit

lab0 `.gdbinit`

```
target remote 127.0.0.1:1234
layout split
```

Warning

Linux `.` `ls` `ls -a`

`loongarch32r-linux-gnusf-gdb start.elf` gdb gdb `.gdbinit` gdb gdb `exit` shell `/`

```
echo "set auto-load safe-path /" > ~/.gdbinit
```

`loongarch32r-linux-gnusf-gdb start.elf` QEMU -s GDB RSP 1234 QEMU

Info

Tips: b [target] .gdbinit

2.6

2.6.1

1.
2.
3. gdb

2.7 ucore

ucore - shell: bash shell -- ls cd rm pwd... - apt git - apt debian, ubuntu linux - git - eclipse-CDT understand gedit vim - Eclipse-CDT Eclipse C/C++ qemu Debug uCore OS - Understand Windows sourceinsight - gedit Linux Windows notepad - vim: Linux/unix emacs exuberant-ctags cscope - diff meld , patch - diff, patch - meld kdiff3 diffmerge P4merge - gcc gdb make - gcc C - gdb - ld - objdump ELF - nm - readelf ELF - make make makefile make - dd - qemu -- qemu CPU LoongArch32 - markdown (ucore_docs) - haroopad - gitbook

2.8

- apt-get
- http://wiki.ubuntu.org.cn/Apt-get%E4%BD%BF%E7%94%A8%E6%8C%87%E5%8D%97
- git github
- http://www.cnblogs.com/cspku/articles/Git_cmds.html
- http://www.worldhello.net/gotgithub/index.html
- diff patch
- http://www.ibm.com/developerworks/cn/linux/l-diffp/index.html
- http://www.cnblogs.com/itech/archive/2009/08/19/1549729.html
- gcc
- http://wiki.ubuntu.org.cn/Gcchowto
- http://wiki.ubuntu.org.cn/Compiling_Cpp
- http://wiki.ubuntu.org.cn/C_Cpp_IDE
- http://wiki.ubuntu.org.cn/C%E8%AF%AD%E8%A8%80%E7%AE%80%E8%A6%81%E8%AF%AD%E6%B3%95%E6%8C%87%E5%8D%97
- gdb
- http://wiki.ubuntu.org.cn/%E7%94%A8GDB%E8%B0%83%E8%AF%95%E7%A8%8B%E5%BA%8F
- make & makefile
- http://wiki.ubuntu.com.cn/index.php?title=%E8%B7%9F%E6%88%91%E4%B8%80%E8%B5%B7%E5%86%99Makefile&variant=zh-cn
- http://blog.csdn.net/a_ran/article/details/43937041
- shell
- http://wiki.ubuntu.org.cn/Shell%E7%BC%96%E7%A8%8B%E5%9F%BA%E7%A1%80
- http://wiki.ubuntu.org.cn/%E9%AB%98%E7%BA%A7Bash%E8%84%9A%E6%9C%AC%E7%BC%96%E7%A8%8B%E6%8C%87%E5%8D%97
- understand
- http://blog.csdn.net/qwang24/article/details/4064975
- vim
- http://www.httpy.com/html/wangluobiancheng/Perljiaocheng/2014/0613/93894.html
- http://wenku.baidu.com/view/4b004dd5360cba1aa811da77.html
- meld
- https://linuxtoy.org/archives/meld-2.html
- qemu

- http://wenku.baidu.com/view/04c0116aa45177232f60a2eb.html
- Eclipse-CDT
- http://blog.csdn.net/anzhu_111/article/details/5946634
- haroopad
- http://pad.haroopress.com/
- gitbook
- https://github.com/GitbookIO/gitbook https://www.gitbook.com/

2.9

http://pdos.csail.mit.edu/6.828/2014/reference.html

UNIX general info

- Youtube Unix intro
- The UNIX Time-Sharing System, Dennis M. Ritchie and Ken L.Thompson,. Bell System Technical Journal 57, number 6, part 2 (July-August 1978) pages 1905-1930.
- The Evolution of the Unix Time-sharing System, Dennis M. Ritchie, 1979.
- The C programming language (second edition) by Kernighan and Ritchie. Prentice Hall, Inc., 1988. ISBN 0-13-110362-8, 1998.

building or reading a small OS

- How to make an Operating System
- xv6 book
- , ,2005
- Linux-0.11 2009
- oldlinux
- osdev.org

some OS course

- 6.828: Operating Systems Engineering - in MIT
- CS-537: Introduction to Operating Systems - in WISC

16550 UART Serial Port

- PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs, National Semiconductor, 1995.
- http://byterunner.com/16550.html, Byterunner Technologies.
- Interfacing the Serial / RS232 Port,, Craig Peacock, August 2001.

3. 1

3.1

3.1.1

lab1 OS lab1 OS OS

3.1.2

LoongArch32 BIOS PMON BIOS ELF uCore

QEMU `-kernel` ELF ELF

- ucore OS
- ucore OS
- ucore OS
- ucore OS
-
-
-

3.2

3.2.1

BIOS

IO
RAM

/ EPROM ROM Flash CPU
IO

LoongArch32 BIOS BIOS ChipLab PMON2000 BIOS tftp ELF
ELF

QEMU BIOS QEMU `-kernel` ELF BIOS ELF

bootloader ucore ucore kern/start.S start ucore ucore
ucore

- DMW `kern/init/entry.S` C `kern/init/init.c`
-
-
-
- while 1

LoongArch32 - Logical Address, - Physical Address,

(1) int val=100; int * point=&val

point

(2) CPU “ ” “ ” CPU “ ”
CPU

CPU LoongArch32 CPU PALEN 0x1f000000~0x1fffffff QEMU
256M

```
+------------------+  <- 0xFFFFFFFF (4GB)
|                |
+------------------+  <- 0x1FFFFFFF (512M)
|   IO          |
+------------------+  <- 0x1F000000 (496M)
|                |
+------------------+  <- 0x0FFFFFFF (256M)
|                |
+------------------+  <- 0x00000000
```

6 LoongArch32

(3)

LoongArch32

- CSR.CRMD DA=1 PG=0 = PALEN 0 PALEN
- CSR.CRMD DA=0 PG=1
- DMW DMW
- TLB TLB TLB TLB

CPU CPU CPU“ ” (polling) CPU CPU “ ” /

LoongArch32 (Exception) uCore

- (EX_IRQ,CSR.ESTAT.Ecode=0)
- Load (EX_TLBL,CSR.ESTAT.Ecode=1)
- Store (EX_TLBS,CSR.ESTAT.Ecode=2)
- TLB (EX_TLBR,CSR.ESTAT.Ecode=31)
- (EX_RI,CSR.ESTAT.Ecode=13)
- (EX_IPE,CSR.ESTAT.Ecode=14)
- (EX_SYS,CSR.ESTAT.Ecode=11)
- (EX_ADE,CSR.ESTAT.Ecode=8)

LoongArch32 TLB TLB

QEMU **CSR.RFBASE** **3** **TLB** **CSR.CRMD**

CSR CSR.EBASE CSR.RFBASE - CSR.CRMD PLV IE CSR.PRMD PPLV IE CSR.CRMD PLV 0 IE 0 - PC CSR.ERA - TLB CSR.RFBASE CSR.EBASE

PC

ERTN - CSR.PRMD PPLV PIE CSR.CRMD PLV IE - CSR.ERA

LAB1

(1)

Lab1 serial_init /kern/driver/console.c

```
......
//     1
outb(COM1 + COM_IER, COM_IER_RDI);
......
//           1
pic_enable(IRQ_COM1);
```

CPU CPU CPU CPU CPU clock_init kern/driver/clock.c

```
    ......
unsigned long timer_config;
unsigned long period = 200000000;
period = period / HZ;
timer_config = period & LISA_CSR_TMCFG_TIMEVAL;
timer_config |= (LISA_CSR_TMCFG_PERIOD | LISA_CSR_TMCFG_EN);
__lcsr_csrwr(timer_config, LISA_CSR_TMCFG);
pic_enable(TIMER0_IRQ);
```

(2)

LoongArch32 `kern/init/init.c` `setup_exception_vector` `__exception_vector` `kern/trap/vectors.S` `kern/trap/exception.S` `ramExcHandle_general`

(3)

trap trap.c `ramExcHandle_general` (exception.S)
trap

1) CPU - CSR.CRMD PLV IE CSR.PRMD PPLV IE CSR.CRMD PLV 0 IE 0 - PC CSR.ERA
- TLB CSR.RFBASE CSR.EBASE

2) `exception.S` `ramExcHandle_general`

CSR KS0 KS1

trapfame `kern/trap/loongarch_trapframe.h`

3) `kern/trap/trap.c` `loongarch_trap` C

`kern/trap/trap.c`

4) `loongarch_trap` trapframe `exception.S` ertn

lab1

3.3

3.3.1

Lab1 C `asm` C CSR C

GCC

GCC inline asm statements basic inline asm statement) extended inline asm statement GCC

```
asm("statements");
```

```
asm("nop");
```

"asm" "__asm__" "\n\t" “\n” "\t” tab gcc asm GCC

```
asm( "li.w $t0, 1\n\t"
    "li.w $t1, 2\n\t"
    "addi.w $t0, $t1, $t2"
);
```

gcc asm(...) " "

t0 t1 gcc GAS GAS t0 t1 t0 t1 _boo GCC

- GCC Manual 5.0.0 pre-release,6.43 How to Use Inline Assembly Language in C Code - GCC-Inline-Assembly-HOWTO

GCC

GCC

```
#define read_a0() ({ \
unsigned int __dummy; \
__asm__( \
    "move $a0, %0\n\t" \
    :"=r" (__dummy)); \
__dummy; \
})
```

GCC

```
asm [volatile] ( Assembler Template
   : Output Operands
   [ : Input Operands
   [ : Clobbers ] ])
```

__asm__ __volatile__ “asm” gcc asm volatile asm volatile(...) __asm__ __volatile__(...) " "movl %%cr0,%0\n\t" “ “ 1 2 CPU LoongArch32 32 gcc

output operand list constraint , “ ”

```
    :"=r" (__dummy)
```

“ r” %0 __dummy

```
:“ m”(__dummy)
```

“ m” __dummy

m, v, o
R
I, h
G
I 0 31

input operand list “ ” “1” “2” clobber list, : “memory” “ ” “memory”

- GCC Manual 5.0.0 pre-release,6.43 How to Use Inline Assembly Language in C Code - GCC-Inline-Assembly-HOWTO

3.4

Makefile Makefile LAB1 := -DLAB1_EX4 -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT(6)

```
LAB1    := -DLAB1_EX4 -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
# LAB2  := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
# LAB3  := -DLAB3_EX1 -DLAB3_EX2
# LAB4  := -DLAB4_EX1 -DLAB4_EX2
# LAB5  := -DLAB5_EX1 -DLAB5_EX2
# LAB6  := -DLAB6_EX2
# LAB7  := -DLAB7_EX1 #-D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

-D_SHOW_100_TICKS "100 ticks" -D_SHOW_SERIAL_INPUT

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA001F000 (phys)
  edata 0xA0151820 (phys)
  end   0xA0154B00 (phys)
Kernel executable memory footprint: 1239KB
LAB1 Check - Please press your keyboard manually and see what happend.
100 ticks
100 ticks
100 ticks
100 ticks
100 ticks
100 ticks
100 ticks
got input
100 ticks
got input s
got input d
got input g
100 ticks
got input s
got input g
100 ticks
```

3.5

3.5.1

lab1 lab1 4 “LAB1” challenge “LAB1” “YOUR CODE”

- markdown - - ucore_lab - OS - OS

1 make

OS

1. ucore-kernel.elf (Makefile)

Makefile

make make

make Makefile `V=@` `@` Makefile `make` `V=`

```
$ make clean
$ make "V="
```

make

```
$ man make
```

2 qemu lab1

qemu gdb

1. uCore
2.

qemu qemu gdb qemu gdb

gdb qemu qemu gdb 1234 qemu gdb

```
target remote localhost:1234
```

qemu qemu gdb

qemu make qemu make debug qemu gdb

gdb

gdb b *[ ] qemu cpu gdb

gdb

gdb next, nexti, step, stepi “ ”

```
next
nexti
step

stepi
```

3 CSR

** “LoongArch32 ”** kern/init/entry.S DMW - DMW - MMIO DMA - LoongArch32 DMW

4

1. LoongArch32 LoongArch32
2. `kern/driver/clock.c` `clock_int_handler` trap 100 kprintf ”100 ticks”
3. `kern/driver/console.c` `serial_int_handler` kprintf

2 3 1 2 3 1 ”100 ticks”

“ ”

4. 2

4.1

4.1.1

-
-
-

4.1.2

cache

4.1.3

ucore

kern_init lab1 pmm_init lab2 intr_enable lab1

free used reserved
ucore kernel TLB Page Page Frame CPU

GDB ucore pmm_init

1. pmm_manager
2. page 4KB
3.
4.
5.
6.

- Page mm/memlayout.h pages page_init
- pmm_manager
- boot_map_segment get_pte

4.2

4.2.1

DMW+

-
-
-
-
-
-

lab1 lab2 MMU [PALEN-1:0] 0 MMU

lab1 lab2 ucore kernel_entry entry.S CSR.CRMD DA=0 PG=1 MMU CSR.DWM0 0xa0000001 0xa0000000-0xbfffffff 0x00000000-0x1fffffff CSR.DWM1 0x80000011 0x80000000-0x9fffffff 0x00000000-0x1fffffff

tools/kernel.ld

```
OUTPUT_ARCH(loongarch)
ENTRY(kernel_entry)
SECTIONS
{
    . = 0xa0000000;

  .text      :
  {
    . = ALIGN(4);
    wrs_kernel_text_start = .; _wrs_kernel_text_start = .;
    *(.startup)
    *(.text)
    *(.text.*)
    *(.gnu.linkonce.t*)
    *(.mips16.fn.*)
    *(.mips16.call.*) /* for MIPS */
    *(.rodata) *(.rodata.*) *(.gnu.linkonce.r*) *(.rodata1)
    . = ALIGN(4096);
    *(.ramexv)
  }
```

ld ucore 0xa0000000 0x00000000

```
 phy addr  = CSR.DMW0[31:29]  : virtual addr[28:0]
```

** ** kernel_entry pmm_init ucore 0x80000000-0x9fffffff 0xa0000000-0xbfffffff 512M

pmm_init pmm_init boot_pgdir

uCore(LoongArch32) 32M x86 e820 RISC ACPI (Device Tree) ARM RISC-V Linux 32M

4KB 4KB Page
Page Page Page kern/mm/memlayout.h kern/default_pmm.c

Page flags Page page_link

```
struct Page {
    int ref;        // page frame's reference counter
    uint32_t flags; // array of flags that describe the status of the page frame
    unsigned int property; // used in buddy system, stores the order (the X in 2^X) of the continuous memory block
    int zone_num;  // used in buddy system, the No. of zone which the page belongs to
    list_entry_t page_link;// free list link
    list_entry_t swap_link;  // swap hash link
};
```

Page ref “ ” Page Page ref
Page ref flags kern/mm/memlayout.h

```
/* Flags describing the status of a page frame */
#define PG_reserved                 0       // the page descriptor is reserved for kernel or unusable
#define PG_property                 1       // the member 'property' is valid
#define PG_slab                     2       // page frame is included in a slab
#define PG_dirty                    3       // the page has been modified
#define PG_swap                     4       // the page is in the active or inactive page list (and swap hash table)
#define PG_active                   5       // the page is in the active page list
```

flags bit bit 0 reserved bit 0 1
“ ” bit 1 free 1 free 0

PG_property best fit, buddy system PG_property

Page property Page Head Page
property property property

Page page_link lab0 Page
Head Page page_link

free_area_t list_entry nr_free

```
/* free_area_t - maintains a doubly linked list to record free (unused) pages */
typedef struct {
            list_entry_t free_list;                                // the list header
            unsigned int nr_free;                                 // # of free pages in this free list
} free_area_t;
```

ucore

- Page
-

memlayout.h KERNBASE 0xa0000000 KMEMSIZE 512M KERNTOP memlayout.h
maxpa KERNTOP page_init Page size 4096 bytes 2^12 bytes npage

```
#define KERNTOP             (KERNBASE + KMEMSIZE)
maxpa = KERNTOP
npage = KMEMSIZE >> PGSHIFT    # PGSHIFT = 12
```

Page

```
sizeof(struct Page) * npage
```

end end Page

```
pages = (struct Page *)ROUNDUP_2N((void *)end, PGSHIFT);
```

0 pages+ sizeof(struct Page) *npage) 0~640KB pages+sizeof(struct Page) *npage)

```
uintptr_t freemem = PADDR((uintptr_t)pages + sizeof(struct Page) * npage);
```

```
for (i = 0; i < npage; i ++) {
SetPageReserved(pages + i);
}
```

```
//          begin     end
...
init_memmap(pa2page(mbegin), (mend - mbegin) >> PGSHIFT );
```

SetPageReserved Page flags PG_reserved init_memmap Page flags ref free_area.free_list

default_pmm.c default_alloc_pages default_free_pages

```
struct pmm_manager {
    const char *name;                                 // XXX_pmm_manager's name
    void (*init)(void);                               // initialize internal description&management data structure
                                                      // (free block list, number of free block) of XXX_pmm_manager
    void (*init_memmap)(struct Page *base, size_t n); // setup description&management data structcure according to
                                                      // the initial free physical memory space
    struct Page *(*alloc_pages)(size_t n);            // allocate >=n pages, depend on the allocation algorithm
    void (*free_pages)(struct Page *base, size_t n);  // free >=n pages with "base" addr of Page descriptor structures(memlayout.h)
    size_t (*nr_free_pages)(void);                    // return the number of free pages
    void (*check)(void);                              // check the correctness of XXX_pmm_manager
};
```

init_memmap/ alloc_pages/free_pages

LoongArch32 MIPS uCore x86 x86 default_pmm_manager 4KB alloc_page Page Directory Table PDT ucore Page Table PT 4KB

0~16MB Page Frame 4KB 4096 4 4096 Page Directory Entry PDE Page Table Entry PTE 4B 4 4KB 4096 16KB 4096*4B 4 16KB 16MB 16MB 5 20KB

0~KERNSIZE ucore KERNSIZE 512MB 131072

1. boot_pgdir
2. boot_map_segment

```
linear addr = phy addr + 0xa0000000
```

32bit la 32bit pa la 10 PTE_P 0 alloc_page 4096

```
         = (         & ~0x0FFF) | PTE_U | PTE_W | PTE_P
```

la 10

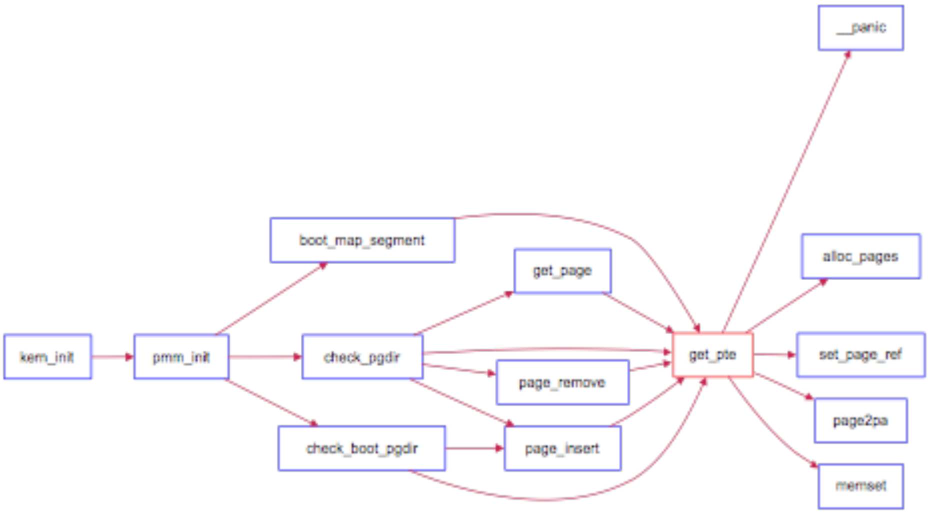
```
        = (pa & ~0x0FFF) | PTE_P | PTE_W
```

- PTE_U 3
- PTE_W 2
- PTE_P 1

ucore get_pte

```
pte_t *get_pte(pde_t *pgdir, uintptr_t la, bool create)
```

get_pte



6 get_pte

pte_t pde_t uintptr_t mm/mmlayout.h libs/types.h unsigned int

pde_t page directory entry pgdir pgd_t pte_t page table entry uintptr_t

pgdir boot_pgdir

create create 0 get_pte NULL create 0 get_pte alloc_page mm/pmm.h

PTE_U PTE_W PTE_P mm/mmu.h

Page ref ref 0 free page_insert page_insert ucore page_remove page_insert

gdt_init

pmm_init ucore

```
Virtual memory map:                                          Permissions
                                                                                  kernel/user

     4G ------------------> +-------------------------------+
                            |                               |
                            |         Mapped(2G)            |
                            |                               |
     KERNTOP  ------------> +-------------------------------+ 0xBFFF_FFFF
                            |                               |
                            |  Unmapped cached(DMW0 512M)   |
                            |                                 |
        KERNBASE------------> +---------------------------------+ 0xa0000000
                            |                                 |
                            |    Unmapped uncached(DMW1 512M) | RW/-- KMEMSIZE
                            |                                 |
                            +---------------------------------+ 0x80000000
                            |                                 |
                            |    User Mapped                  |
                            |                                 |
                            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ 0x00000000
```

ucore FirstFit

lab2 first_fit FirstFit ucore ucore

first_fit

kernel/include/list.h / /

kern/mm/memlayout.h free_area_t

```
    list_entry_t free_list;         // the list header
    unsigned int nr_free;           // # of free pages in this free list
```

buddy_pmm.c free_area

kern/mm/pmm.h pmm_manager init free_area ,first_fit buddy_init init_memmap

```
kern_init --> pmm_init-->page_init-->init_memmap--> pmm_manager->init_memmap
```

default_init_memmap page_init page_init
free_area.free_list Page base->page_link Page page_link

default_init_memmap default_init_memmap

```
default_init_memmap(struct Page *base, size_t n) {
    assert(n > 0);
    struct Page *p = base;
    for (; p != base + n; p ++) {
        assert(PageReserved(p));
        p->flags = p->property = 0;
        set_page_ref(p, 0);
    }
    base->property = n;
    SetPageProperty(base);
    nr_free += n;
    list_add_before(&free_list, &(base->page_link));
}
```

default_alloc_pages n n

```
if (n > nr_free) {
return NULL;
}
```

assert n 0

```
assert(n > 0);
```

n<=0 ucore firstfit list_next le2page Page p p->property
>=n <n list_next list_next== &free_list page default_alloc_pages

```
default_alloc_pages(size_t n) {
    assert(n > 0);
    if (n > nr_free) {
```

```
        return NULL;
    }
    struct Page *page = NULL;
    list_entry_t *le = &free_list;
    // TODO: optimize (next-fit)
    while ((le = list_next(le)) != &free_list) {
        struct Page *p = le2page(le, page_link);
        if (p->property >= n) {
            page = p;
            break;
        }
    }
    if (page != NULL) {
        if (page->property > n) {
            struct Page *p = page + n;
            p->property = page->property - n;
            SetPageProperty(p);
            list_add_after(&(page->page_link), &(p->page_link));
        }
        list_del(&(page->page_link));
        nr_free -= n;
        ClearPageProperty(page);
    }
    return page;
}
```

default_free_pages default_alloc_pages lab2/kernel/mm/ default_pmm.c

4.3

Makefile Makefile LAB2 := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3(7)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
# LAB3  := -DLAB3_EX1 -DLAB3_EX2
# LAB4  := -DLAB4_EX1 -DLAB4_EX2
# LAB5  := -DLAB5_EX1 -DLAB5_EX2
# LAB6  := -DLAB6_EX2
# LAB7  := -DLAB7_EX1 #-D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0020000 (phys)
  edata 0xA0153370 (phys)
  end   0xA0156650 (phys)
Kernel executable memory footprint: 1242KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0197000
free pages: 00001E69
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
LAB2 Check Pass!
```

ucore entry etext edata end ucore page

4.4

4.4.1

lab2 lab2 3 2 “LAB2” challenge “LAB2” “YOUR CODE”

- markdown - - ucore_lab -
OS - OS

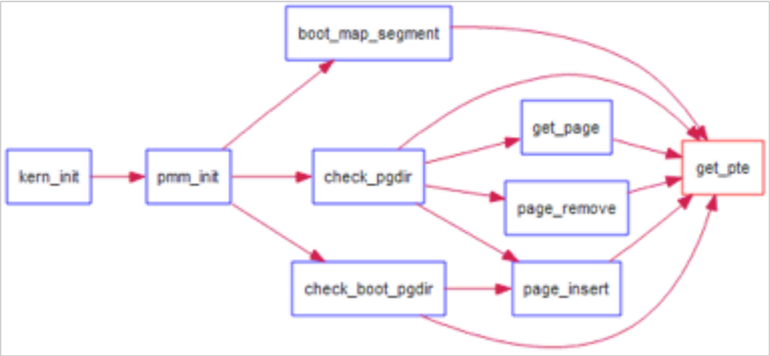
1 first-fit

first fit : default_pmm.c default_init
default_init_memmap default_alloc_pages default_free_pages default_pmm.c

- first fit

2

get_pte
get_pte kern/mm/pmm.c get_pte get_pte

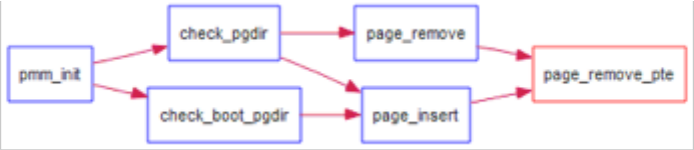


1 get_pte

- Page Directory Entry Page Table Entry ucore - ucore

3

Page page_remove_pte
kern/mm/pmm.c page_remove_pte page_remove_pte



2 page_remove_pte

- Page - lab2

Challenge buddy system

Buddy System (Block) , 2 n (Pow(2, n)), 1, 2, 4, 8, 16, 32, 64, 128...

- ucore buddy system

Challenge slub

slub

- linux slub / ucore slub

Challenges Challenge Challenge

5. 3

5.1

TLB Refill Page Fault

5.1.1

- TLB
- LoongArch32 TLB
- Page Fault

5.1.2

TLB TLB

5.2

5.2.1

CPU“ ”

1.

2.

3. CPU

“ ” CPU“ ”

demand paging CPU page swap in/out LoongArch32 uCore swap “ ”

ucore lab3

ucore init vmm_init setup_exception_vector pic_init lab1 pmm_init lab2

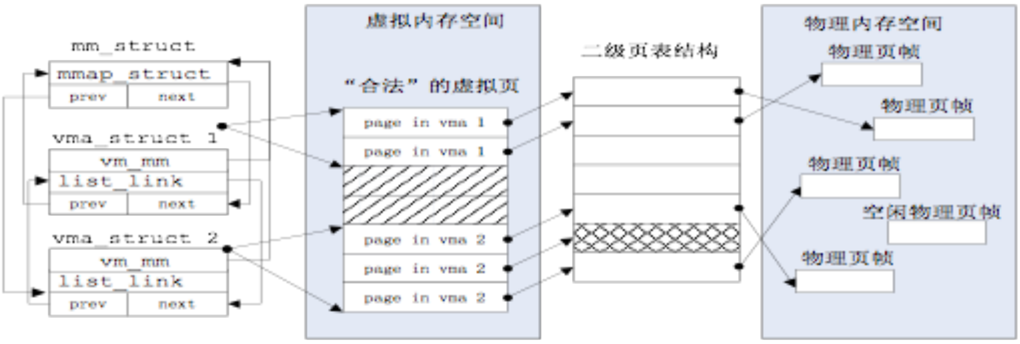
pmm_init lab3 vmm_init 2 1 TLB vmm_init TLB `kern/mm/la32_tlb.c` LoongArch32 “ ” “ ” LoongArch32 TLB CSR TLB 1

ucore init vmm_init 1 2 “ ” ucore mm_struct vma_struct “ ” ucore “ ” 2 do_pgfault “ ” 1

lab3

-- “ ” “ ” “ ” “ ” ucore page fault

page_fault “ ” ucore “ ” ucore mm_struct vma_struct ucore page fault ucore vma_struct mm_struct vma_struct



ucore “ ” vma_struct vmm.h vma_struct vma_struct vma vma_struct

```
struct vma_struct {
    // the set of vma using the same PDT
    struct mm_struct *vm_mm;
    uintptr_t vm_start; // start addr of vma
    uintptr_t vm_end; // end addr of vma
    uint32_t vm_flags; // flags of vma
    //linear list link which sorted by start addr of vma
    list_entry_t list_link;
};
```

vm_start vm_end PGSIZE vm_start < vm_end list_link vma_struct vma_struct vma vm_flags

```
#define VM_READ 0x00000001 //
#define VM_WRITE 0x00000002 //
#define VM_EXEC 0x00000004 //
```

vm_mm vma_struct mm_struct mm_struct mm

```
struct mm_struct {
    // linear list link which sorted by start addr of vma
    list_entry_t mmap_list;
    // current accessed vma, used for speed purpose
    struct vma_struct *mmap_cache;
    pde_t *pgdir; // the PDT of these vma
    int map_count; // the count of these vma
};
```

mmap_list mmap_cache “ ” mmap_cache mm_struct 30% pgdir mm_struct pgdir map_count mmap_list vma_struct

vma_struct

- vma_create-- vma
- insert_vma_struct-- vma
- find_vma-- vma

vma_create vm_start vm_end vm_flags vma_struct insert_vma_struct vma [vma->vm_start,vma->vm_end] mm mmap_list find_vma addr mm mm mmap_list vma addr vma->vm_start<=addr <vma->end page fault

mm_struct mm_create mm_destroy mm_struct mm_create kmalloc mm_destroy ucore vma_struct mm_destroy mmap_list vma

TLB Refill

LoongArch32 TLB -- handle_tlbmiss TLB CPU TLB Refill CSR.PRMD TLB Refill TLB Refill uCore TLB Refill CRMD trap_dispatch trapframe CSR.ESTAT TLB Refill `handle_tlbmiss`

loongarch_trap-->trap_dispatch-->handle_tlbmiss

`handle_tlbmiss` PTE Page Fault `pgfault_handler` `do_pgfault` do_pgfault do_pgfault

loongarch_trap-->trap_dispatch-->handle_tlbmiss-->pgfault_handler-->do_pgfault

ucore do_pgfault trapframe CPU CSR.BADVA errorCode VMA TLB VMA

5.3

Makefile Makefile LAB3 := -DLAB3_EX1 -DLAB3_EX2(8)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
LAB3    := -DLAB3_EX1 -DLAB3_EX2
# LAB4  := -DLAB4_EX1 -DLAB4_EX2
# LAB5  := -DLAB5_EX1 -DLAB5_EX2
# LAB6  := -DLAB6_EX2
# LAB7  := -DLAB7_EX1 #-D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
((THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0020000 (phys)
  edata 0xA0153A20 (phys)
  end   0xA0156D00 (phys)
Kernel executable memory footprint: 1244KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0197000
free pages: 00001E69
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
LAB3 Check Pass!
```

5.4

lab3 lab2 2 “LAB3” challenge “LAB3” “YOUR CODE” - markdown - - ucore_lab - OS - OS

1 TLB

LoongArch32 LoongArch32 TLB TLB TLBELO TLBEHI bit `kern/mm/la32_tlb.c`

2

`pte2tlblow` uCore PTE TLBELO

`tlb_refill` LoongArch32 13 0 1 `pte2tlblow` TLBELO TLBEHI `tlb_replace_random` TLB

2

do_pgfault mm/vmm.c VMA LAB3 EXERCISE 2

```
make qemu
```

check_pgfault “check_pgfault()succeeded!” 1 2

- Page Directory Entry Page Table Entry ucore
- ucore

6. 4

6.1

6.1.1

- /
-

6.1.2

⅔ ucore OS CPU “ ” “ ” “ ” CPU

-
-
- ucore
-

/

6.1.3

scheduler CPU CPU lab4

kern/init/init.c kern_init kern_init proc_init idleproc initproc idleproc cpu_idle idleproc ucore kernel_thread initproc initproc “Hello World”

lab4 cpu_idle idleproc need_resched 1 idleproc 1 schedule “ ” “ ” “PROC_RUNNABLE” switch_to ( ) initproc

-- proc_struct ucore idleproc initproc initproc

6.2

6.2.1

proc.c alloc_proc / “ ” uCore OS boot_cr3 uCore OS uCore OS “ ”

--

struct proc_struct *kern/process/proc.h*

```
struct proc_struct {
    enum proc_state state;                      // Process state
    int pid;                                    // Process ID
    int runs;                                   // the running times of Proces
    uintptr_t kstack;                           // Process kernel stack
    volatile bool need_resched;                 // bool value: need to be rescheduled to release CPU?
    struct proc_struct *parent;                 // the parent process
    struct mm_struct *mm;                       // Process's memory management field
    struct context context;                     // Switch here to run process
    struct trapframe *tf;                       // Trap frame for current interrupt
    uintptr_t cr3;                              // the base addr of Page Directroy Table(PDT)
    uint32_t flags;                             // Process flag
    char name[PROC_NAME_LEN + 1];               // Process name
    list_entry_t list_link;                     // Process link list
    list_entry_t hash_link;                     // Process hash list
    int exit_code;                              // exit code (be sent to parent proc)
    uint32_t wait_state;                        // waiting state
    struct proc_struct *cptr, *yptr, *optr;     // relations between processes
    struct run_queue *rq;                       // running queue contains Process
    list_entry_t run_link;                      // the entry linked in run queue
    int time_slice;                             // time slice for occupying the CPU
    struct fs_struct *fs_struct;                // the file related info(pwd, files_count, files_array, fs_semaphore) of process
};
```

- mm mm lab3 OS swap page lab5 swap page mm lab4 mm proc_struct *mm=0 mm pgdir *mm=NULL proc_struct pgdir proc_struct cr3
- state
- parent idleproc
- context switch.S uCore context context *kern/process/switch.S* switch_to
- tf uCore tf trapframe uCore tf trap.c::trap
- cr3: cr3 x86 CSR x86 TLB LoongArch32 uCore uCore lcr3 mm cr3 mm mm NULL PCB cr3 mm pgdir cr3 boot_cr3 boot_cr3 uCore
- kstack: uCore 2 memlayout.h KSTACKSIZE kstack / kstack tss tss mm kstack uCore linux linux linux kernel

uCore *kern/process/proc.c*

- static struct proc *current CPU “ ” switch_to
- static struct proc *initproc
- static list_entry_t hash_list[HASH_LIST_SIZE] proc_struct hash_link pid
- list_entry_t proc_list proc_struct list_link

0 idleproc

init.c::kern_init proc.c::proc_init proc_init kern_init uCore uCore 0 -- idleproc

alloc_proc kmalloc proc_struct - 0 proc proc_struct

```
proc->state = PROC_UNINIT;       “  ”
proc->pid = -1;                 pid
proc->cr3 = boot_cr3;
...
```

, “ ” “ ” pid -1 “ ” uCore uCore boot_cr3 “ ” “ ”—uCore

proc_init idleproc

```
idleproc->pid = 0;
idleproc->state = PROC_RUNNABLE;
idleproc->kstack = (uintptr_t)bootstack;
idleproc->need_resched = 1;
set_proc_name(idleproc, "idle");
```

4 idleproc --0 idleproc 0 pid “0” C “0” idleproc “ ” “ ” uCore idleproc uCore idleproc uCore CPU idleproc “ ” idleproc->need_resched “1” idleproc --cpu_idle idleproc 1 schedule

1 initproc

0 cpu_idle uCore idleproc kernel_thread init_main init_main init_main kernel_thread

```
kernel_thread(int (*fn)(void *), void *arg, uint32_t clone_flags)
{
     struct trapframe tf;
    memset(&tf, 0, sizeof(struct trapframe));
    tf.tf_regs.reg_r[LOONGARCH_REG_A0] = (uint32_t)arg;
    tf.tf_regs.reg_r[LOONGARCH_REG_A1] = (uint32_t)fn;
    tf.tf_regs.reg_r[LOONGARCH_REG_A7] = 0;
    tf.tf_estat = read_csr_exst();
    tf.tf_era = (uint32_t)kernel_thread_entry;
    return do_fork(clone_flags | CLONE_VM, 0, &tf);
}
```

kernel_thread tf do_fork do_fork copy_thread tf tf.tf_regs.reg_r[LOONGARCH_REG_A0],tf.tf_regs.reg_r[LOONGARCH_REG_A1],tf.tf_regs.reg_r[LOONGARCH_REG_V7] tf.tf_estat tf.tf_era kernel_thread_entry kern/process/entry.S kernel_thread_entry entry.S

```
kernel_thread_entry:
    addi.w  sp, sp,  -16
    //goto kernel_thread
    addi.w  t0, a1, 0
  //  la.abs  t0, a1
    jirl    ra, t0, 0
   // bl a1
```

```
    move    v0, a0
    //goto do_exit():see proc.c
    la.abs  t0, do_exit
    jirl    ra, t0, 0
```

kernel_thread_entry fn “ ” fn arg a0 fn a0 do_exit

do_fork kernel_thread do_fork do_fork 2 do_fork 6

1. alloc_proc
2. setup_stack
3. clone_flag copy_mm
4. copy_thread
5. hash_list proc_list
6. “ ”
7. id

3 copy_mm current->mm NULL proc->mm mm copy_thread

```
static void
copy_thread(struct proc_struct *proc, uintptr_t esp, struct trapframe *tf) {
    proc->tf = (struct trapframe *)(proc->kstack + KSTACKSIZE) - 1;
    *(proc->tf) = *tf;
    proc->tf->tf_regs.reg_r[LOONGARCH_REG_A7] = 0; // use A7 as syscall result register
    if(esp == 0) //a kernel thread
      esp = (uintptr_t)proc->tf - 32;
    proc->tf->tf_regs.reg_r[LOONGARCH_REG_SP] = esp;
    proc->context.sf_ra = (uintptr_t)forkret;
    proc->context.sf_sp = (uintptr_t)(proc->tf) - 32;
}
```

//////////////////////// kernel_thread sp initproc

```
//////////////////////////
//
initproc->tf= (proc->kstack+KSTACKSIZE) – sizeof (struct trapframe);
//
initproc->tf.tf_regs.reg_r[LOONGARCH_REG_A0] = (uint32_t)init_main;
initproc->tf.tf_regs.reg_r[LOONGARCH_REG_A1] = (uint32_t)fn;
initproc->tf.tf_regs.reg_r[LOONGARCH_REG_A7] = 0;
initproc->tf.tf_era = (uint32_t)kernel_thread_entry;
initproc->tf->tf_regs.reg_r[LOONGARCH_REG_SP] = esp;
initproc->context.sf_ra = (uintptr_t)forkret;
initproc->context.sf_sp = (uintptr_t)(initproc->tf) - 32;
```

initproc process context uCore initproc initproc->context initproc initproc context.sf_ra context.sf_sp initproc initproc initproc initproc initproc context.sf_sp initproc context.sf_ra initproc forkret do_fork initproc

initproc

uCore proc_init idleproc initproc uCore idleproc init cpu_idle uCore idleproc cpu_idle CPU

```
void
cpu_idle(void) {
    while (1) {
        if (current->need_resched) {
            schedule();
            ……
```

idleproc need_resched 0 “ idleproc” proc_init idleproc idleproc->need_resched 1 schedule “ ”

uCore FIFO schedule

1 current->need_resched 0 2 proc_list “ ” next 3 proc_run current

next proc10 idleproc CPU initproc schedule proc_list “ ” initproc proc_run switch_to

1. current next initproc
2. current_pgdir next initproc next->cr3
3. switch_to switch_to initproc

idleproc initproc boot_cr3

proc_run switch_to process context “ ” context switch.S switch_to

```
.text
.globl switch_to
switch_to:
//save the registers
    st.w    sp, a0, 48
    st.w    fp, a0, 44
    st.w    ra, a0, 40
    st.w    tp, a0, 36
    st.w    s8, a0, 32
    st.w    s7, a0, 28
    st.w    s6, a0, 24
    st.w    s5, a0, 20
    st.w    s4, a0, 16
    st.w    s3, a0, 12
    st.w    s2, a0, 8
    st.w    s1, a0, 4
    st.w    s0, a0, 0

    //use as nop
    dbar    0
```

switch_to context.sf_ra

```
 st.w    ra, a0, 40
```

10 context context context.sf_ra uCore initproc initproc initproc->context.sf_era = (uintptr_t)forkret switch_to initproc forkret forkret kern/trap/trapentry.S forkrets

```
.global forkrets
.type forkrets, @function
forkrets:
  addi.w a0, sp, -16
  b exception_return
.end forkrets
```

forkrets sp initproc Initprocde kernel_tread_entry do_exit do_exit

6.3

Makefile Makefile LAB4 := -DLAB4_EX1 -DLAB4_EX2(9)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
LAB3    := -DLAB3_EX1 -DLAB3_EX2
LAB4    := -DLAB4_EX1 -DLAB4_EX2
# LAB5  := -DLAB5_EX1 -DLAB5_EX2
# LAB6  := -DLAB6_EX2
# LAB7  := -DLAB7_EX1 #-D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0020000 (phys)
  edata 0xA0153CF0 (phys)
  end   0xA0156FD0 (phys)
Kernel executable memory footprint: 1244KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0197000
free pages: 00001E69
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: RR_scheduler
proc_init succeeded
this initproc, pid = 1, name = "init"
To U: "(null)".
To U: "en.., Bye, Bye. :)"
kernel panic at kern/process/proc.c:1274:
    LAB4 Check Passed!
Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
K>
```

6.4

6.4.1

lab4 lab2 3 1 “LAB4” challenge “LAB4” “YOUR CODE” - markdown - - sys_code - OS - OS

1

alloc_proc kern/process/proc.c struct proc_struct ucore

alloc_proc proc_struct state/pid/runs/kstack/need_resched/parent/mm/context/tf/cr3/flags/name

- proc_struct `struct context context` `struct trapframe *tf`

2

kernel_thread **do_fork** do_kernel alloc_proc alloc_proc ucore do_fork do_fork kern/process/proc.c do_fork

- alloc_proc
-
-
-
-
-
-

- ucore fork id

3 proc_run

proc_run - - `local_intr_save(intr_flag);....local_intr_restore(intr_flag);` ?

make qemu -j 16

Challenge

slab slab slab first-fit/best-fit/worst-fit/buddy

Linux

SLOB

http://en.wikipedia.org/wiki/SLOB http://lwn.net/Articles/157944/

SLAB

https://www.ibm.com/developerworks/cn/linux/l-linux-slab-allocator/

6.5

6.5.1

“ ” “ ” “ ” “ ” ucore

CPU “ ”

CPU CPU CPU “ ” /

1.

CPU

“ ” CPU / CPU CPU

2.

“ ” “ ” “ ” CPU “ ” CPU CPU “ ” CPU CPU CPU “ ” “ ”

“ ” “ ”/“ ” “ ” “ ”

3.

CPU

CPU

ucore ucore ucore ucore / /

7. 5

7.1

7.1.1

-
-
- ucore sys_fork/sys_exec/sys_exit/sys_wait

7.1.2

4 5 ucore ucore sys_fork/ sys_exec/sys_exit/sys_wait

7.1.3

ucore “ ” CPU ucore

lab4 ucore kern_init lab4 lab5 kern_init

lab4 CPU CSR.CRMD PLV 0 CPU 3 CPU CPU copy_from_user copy_to_user CPU

ELF fork CPU

ucore “ ” lab4 proc_init alloc ucore idle kernl_thread init_main init_main user_main user_main kernel_tread kernel_execve ld CPU ucore CPU ucore FIFO CPU ucore

7.2

7.2.1

1.

hello user/hello.c

```
#include <stdio.h>
#include <ulib.h>

int main(void) {
    cprintf("Hello world!!.\n");
    cprintf("I am process %d.\n", getpid());
    cprintf("hello pass.\n");
    return 0;
}
```

hello sys_getpid getpid hello --pid

ucore hello ucore hello make

```
……

+ cc user/hello.c

loongarch32-linux-gnu-gcc -c  -Iuser/libs -Ikern/include -fno-builtin-fprintf -fno-builtin -nostdlib  -nostdinc -g -G0 -Wa,-O0 -fno-pic -mno-shared -msoft-float -ggdb -gstabs -mlcsr   user/hello.c  -o obj/user/hello.o

loongarch32-linux-gnu-ld -T user/libs/user.ld  obj/user/hello.o --whole-archive obj/user/libuser.a -o obj/user/hello

……

sed 's/$FILE/hello/g' tools/piggy.S.in > obj/user/hello.S

……

#        obj/user/hello.S        .incbin         obj/user/hello
loongarch32-linux-gnu-gcc -c obj/user/hello.S -o obj/user/hello.piggy.o

loongarch32-linux-gnu-ld -nostdlib -n -G 0 -static -T tools/kernel.ld  obj/init/init.o  …… obj/user/hello.piggy.o …… -o obj/ucore-kernel-piggy
```

hello hello.c hello

- user/libs/initcode.S “_start” gp sp umain
- user/libs/umain.c umain C main main exit exit sys_exit
- user/libs/ulib.[ch] C
- user/libs/syscall.[ch]
- user/libs/stdio.c cprintf sys_putc
- user/libs/panic.c __panic/__warn sys_exit

libs/*.[ch] UNIX libc hello

libs/*.[ch] user/libs/*.[ch] user/*.[ch]

a00c3160 _binary_obj_user_hello_end a00125ac file_open a00018d0 strcpy a0000240 ide_device_valid a01455c0 _binary_obj_user_forktree_start a000cb30 wakeup_queue a00156f0 vfs_set_bootfs a000d12c cond_signal a0011850 sysfile_fsync a001a57c dev_init_stdout a00b4ac0 _binary_obj_user_hello_start

make ld hello obj/user/hello.piggy.o ucore kernel `tools/piggy.S.in` .global _binary_obj_user_FILE_start .global _binary_obj_user_FILE_end ld hello ucore bootloader hello ucore initrd

2.

user/libs/user.ld

```
SECTIONS {
    /* Load programs at this address: "." means the current address */
    . = 0x10000000;
```

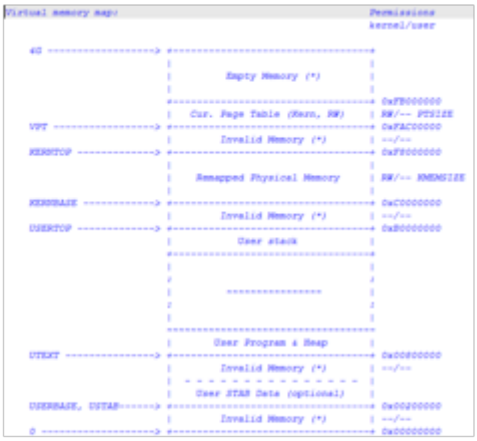
tools/kernel.ld

```
SECTIONS {
    /* Load the kernel at this address: "." means the current address */
    . = 0xa0000000;
```

ucore

ucore “ ”

ucore kern/mm/memlayout.h



```
Virtual memory map:                                          Permissions
                                                             kernel/user

    4G -------------------> +---------------------------------+
                            |                                 |
                            |         Empty Memory (*)        |
                            |                                 |
                            +---------------------------------+ 0xFB000000
                            |   Cur. Page Table (Kern, RW)    | RW/-- PTSIZE
    VPT -----------------> +---------------------------------+ 0xFAC00000
                            |        Invalid Memory (*)       | --/--
    KERNTOP -------------> +---------------------------------+ 0xF8000000
                            |                                 |
                            |    Remapped Physical Memory     | RW/-- KMEMSIZE
                            |                                 |
    KERNBASE ------------> +---------------------------------+ 0xC0000000
                            |        Invalid Memory (*)       | --/--
    USERTOP -------------> +---------------------------------+ 0xB0000000
                            |           User stack            |
                            +---------------------------------+
                            |                                 |
                            :                                 :
                            |         ~~~~~~~~~~~~~~~~        |
                            :                                 :
                            |                                 |
                            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                            |       User Program & Heap       |
    UTEXT ---------------> +---------------------------------+ 0x00800000
                            |        Invalid Memory (*)       | --/--
                            |  - - - - - - - - - - - - - - -  |
                            |    User STAB Data (optional)    |
    USERBASE, USTAB------> +---------------------------------+ 0x00200000
                            |        Invalid Memory (*)       | --/--
    0 -------------------> +---------------------------------+ 0x00000000
```

3.

initproc hello initproc

```
    // kernel_execve - do SYS_exec syscall to exec a user program called by user_main kernel_thread
static int kernel_execve(const char *name, unsigned char *binary, size_t size) {
    int ret, len = strlen(name);
    asm volatile(
      "addi.w   $a7, $zero,%1;\n" // syscall no.
      "move $a0, %2;\n"
      "move $a1, %3;\n"
      "move $a2, %4;\n"
      "move $a3, %5;\n"
      "syscall  0;\n"
      "move %0, $a7;\n"
      : "=r"(ret)
      : "i"(SYSCALL_BASE+SYS_exec), "r"(name), "r"(len), "r"(binary), "r"(size)
      : "a0", "a1", "a2", "a3", "a7"
    );
    return ret;
}

#define __KERNEL_EXECVE(name, path, ...) ({                          \
const char *argv[] = {path, ##__VA_ARGS__, NULL};       \
                     kprintf("kernel_execve: pid = %d, name = \"%s\".\n",    \
                             current->pid, name);                            \
                     kernel_execve(name, argv);                              \
})

#define KERNEL_EXECVE(x, ...)                   __KERNEL_EXECVE(#x, #x, ##__VA_ARGS__)
……
// init_main - the second kernel thread used to create kswapd_main & user_main kernel threads
static int
init_main(void *arg) {
    #ifdef TEST
    KERNEL_EXECVE2(TEST, TESTSTART, TESTSIZE);
    #else
    KERNEL_EXECVE(hello);
    #endif
```

```
    panic("kernel_execve failed.\n");
    return 0;
}
```

/ Initproc init_main KERNEL_EXECVE(hello) kernel_execve SYS_exec ld hello

- _binary_obj___user_hello_out_start hello
- _binary_obj___user_hello_out_size hello

kernel_execve SYS_exec ucore ucore

```
vector128(vectors.S)--\>
\_\_alltraps(trapentry.S)--\>trap(trap.c)--\>trap\_dispatch(trap.c)--
--\>syscall(syscall.c)--\>sys\_exec syscall.c --\>do\_execve(proc.c)
```

do_execve

- mm NULL mm 1 0 0 mm mm initproc mm NULL
- ELF load_icode

load_icode

1. mm_create mm mm
2. setup_pgdir ucore boot_pgdir mm->pgdir
3. ELF mm_map ELF BSS vma vma mm
4.
5. mm_mmap vma 256 1MB ←>
6. , vma mm mm->pgdir current_pgdir initproc hello
7. CPU

initproc “ertn” exception.S hello _start user/libs/initcode.S

ucore

exit sys_exit ucore

exit error_code ucore ucore do_exit

1. current->mm != NULL

a) “lcr3(boot_cr3)”

b) mm mm_count 1 0 mm

i. exit_mmap current->mm->vma vma

ii. put_pgdir

iii. mm_destroy mm vma mm

c) current->mm NULL

2. current->state=PROC_ZOMBIE current->exit_code=error_code

3. current->parent

current->parent->wait_state==WT_CHILD

“wakup_proc(current->parent)”

4. initproc initproc PROC_ZOMBIE initproc

5. schedule()

wait wait_pid wait wait_pid id pid sys_wait ucore

1. pid!=0 id pid

2. PROC_ZOMBIE PROC_SLEEPING WT_CHILD schedule() 1

3. PROC_ZOMBIE proc_list hash_list

System Call “ ” “ ” “ ”

ucore

1.

libc user/libs/ulib.[ch] user/libs/syscall.[ch] syscall

```
static inline int
syscall(int num, ...) {
    va_list ap;
    va_start(ap, num);
    uint32_t arg[MAX_ARGS];
    int i, ret;
    for (i = 0; i < MAX_ARGS; i ++) {
        arg[i] = va_arg(ap, uint32_t);
    }
    va_end(ap);

    num += SYSCALL_BASE;
    asm volatile(
      "move $a7, %1;\n" /* syscall no. */
      "move $a0, %2;\n"
      "move $a1, %3;\n"
      "move $a2, %4;\n"
      "move $a3, %5;\n"
      "syscall 0;\n"
      "move %0, $a7;\n"
      : "=r"(ret)
      : "r"(num), "r"(arg[0]), "r"(arg[1]), "r"(arg[2]), "r"(arg[3])
      : "a0", "a1", "a2", "a3", "a7"
    );
    return ret;
}
```

2.

SYS_exit	process exit	do_exit
SYS_fork	create child process, dup mm	do_fork-->wakeup_proc
SYS_wait	wait child process	do_wait
SYS_exec	after fork, process execute a new program	load a program and refresh the mm
SYS_yield	process flag itself need resecheduling	proc->need_sched=1, then scheduler will rescheule this process
SYS_kill	kill process	do_kill-->proc->flags |= PF_EXITING, -->wakeup_proc-->do_wait-->do_exit
SYS_getpid	get the process's pid	

/

3.

- “syscall”
- “ertn”
-
-

getpid getpid “syscall” CPU exception13 kern/trap/exception.S

```
exception13(exception.S)--\>
\_\loongarch_trap(trap.c)--\>trap\_dispatch(trap.c)--
--\>syscall(syscall.c)--\>sys\_getpid(syscall.c)--\>……--\>\_\exception_return(exception.S)
```

trap trapframe trapframe exception.S exception_handler

```
exception_handler:
    // Save t0 and t1
    csrwr   t0, LISA_CSR_KS0
    csrwr   t1, LISA_CSR_KS1
    // Save previous stack pointer in t1
    move    t1, sp
    csrwr   t1, LISA_CSR_KS2
    //t1 saved the vaual of KS2,KS2 saved sp
    /*
        Warning: csrwr will bring the old csr register value into rd,
        not only just write rd to csr register,
        so you may see the rd changed.
        It's documented in the manual from loongarch.
    */

    // check if user mode
    csrrd   t0, LISA_CSR_PRMD
    andi    t0, t0, 3
    beq     t0, zero, 1f


    /* Coming from user mode - load kernel stack into sp */
    la      t0, current // current pointer
    ld.w    t0, t0, 0 // proc struct
    ld.w    t0, t0, 12 // kstack pointer
    addi.w  t1, zero, 1
    slli.w  t1, t1, 13 // KSTACKSIZE=8192=pow(2,13)
    add.w   sp, t0, t1
    csrrd   t1, LISA_CSR_KS2

1:
    //saved EXST to t0 for save EXST to sp later(line 114)
    csrrd   t0, LISA_CSR_EXST
    //return KS2
    csrrd   t1, LISA_CSR_KS2
    b common_exception


common_exception:
   /*
    * At this point:
    *      Interrupts are off. (The processor did this for us.)
    *      t0 contains the exception status(like exception cause on MIPS).
    *      t1 contains the old stack pointer.
    *      sp points into the kernel stack.
    *      All other registers are untouched.
    */

   /*
    * Allocate stack space for 35 words to hold the trap frame,
    * plus four more words for a minimal argument block.
    */
    addi.w  sp, sp, -156
    st.w    s8, sp, 148
    st.w    s7, sp, 144
    st.w    s6, sp, 140
    st.w    s5, sp, 136
    st.w    s4, sp, 132
    st.w    s3, sp, 128
    st.w    s2, sp, 124
    st.w    s1, sp, 120
    st.w    s0, sp, 116
    st.w    fp, sp, 112
    st.w    reserved_reg, sp, 108
    st.w    t8, sp, 104
    st.w    t7, sp, 100
    st.w    t6, sp, 96
    st.w    t5, sp, 92
    st.w    t4, sp, 88
    st.w    t3, sp, 84
    st.w    t2, sp, 80
    //st.w    t1, sp, 76
    //st.w    t0, sp, 72
    st.w    a7, sp, 68
    st.w    a6, sp, 64
    st.w    a5, sp, 60
    st.w    a4, sp, 56
    st.w    a3, sp, 52
    st.w    a2, sp, 48
    st.w    a1, sp, 44
    st.w    a0, sp, 40
    st.w    t1, sp, 36  // replace sp with real sp, now use t1 for free
    st.w    tp, sp, 32
    // save real t0 and t1 after real sp (stored in t1 previously) stored
    csrrd   t1, LISA_CSR_KS1
    st.w    t1, sp, 76
    csrrd   t1, LISA_CSR_KS0
    st.w    t1, sp, 72

    // replace with real value
    // save tf_era after t0 and t1 saved
    csrrd   t1, LISA_CSR_EPC
    st.w    t1, sp, 152

   /*
    * Save remaining exception context information.
    */

    // save ra (note: not in pushregs, it's tf_ra)
    st.w    ra, sp, 28
    // save prmd
    csrrd   t1, LISA_CSR_PRMD
    st.w    t1, sp, 24
    // save estat
    st.w    t0, sp, 20
    // now use t0 for free
    // store badv
    csrrd   t0, LISA_CSR_BADV
    st.w    t0, sp, 16
    st.w    zero, sp, 12
    // support nested interrupt

    // IE and PLV will automatically set to 0 when trap occur

    // set trapframe as function argument
    addi.w  a0, sp, 16
    li  t0, 0xb0    # PLV=0, IE=0, PG=1
    csrwr   t0, LISA_CSR_CRMD
    la.abs  t0, loongarch_trap
    jirl    ra, t0, 0
    //bl loongarch_trap
……
```

trapframe sys_getpid pid
__exception.S trapframe

```
exception_return:
    // restore prmd
    ld.w    t0, sp, 24
    li      t1, 7
    csrxchg t0, t1, LISA_CSR_PRMD
    // restore era no k0 and k1 for la32, so must do first
    ld.w    t0, sp, 152
    csrwr   t0, LISA_CSR_EPC
    // restore general registers
    ld.w    ra, sp, 28
    ld.w    tp, sp, 32
    //ld.w    sp, sp, 36 (do it finally)
    ld.w    a0, sp, 40
    ld.w    a1, sp, 44
    ld.w    a2, sp, 48
    ld.w    a3, sp, 52
    ld.w    a4, sp, 56
    ld.w    a5, sp, 60
    ld.w    a6, sp, 64
    ld.w    a7, sp, 68
    ld.w    t0, sp, 72
    ld.w    t1, sp, 76
    ld.w    t2, sp, 80
    ld.w    t3, sp, 84
    ld.w    t4, sp, 88
    ld.w    t5, sp, 92
    ld.w    t6, sp, 96
    ld.w    t7, sp, 100
    ld.w    t8, sp, 104
    ld.w    reserved_reg, sp, 108
    ld.w    fp, sp, 112
    ld.w    s0, sp, 116
    ld.w    s1, sp, 120
    ld.w    s2, sp, 124
    ld.w    s3, sp, 128
    ld.w    s4, sp, 132
    ld.w    s5, sp, 136
    ld.w    s6, sp, 140
    ld.w    s7, sp, 144
    ld.w    s8, sp, 148
    // restore sp
    ld.w    sp, sp, 36
    ertn

    .end exception_return
    .end common_exception
```

“ertn” CPU “syscall”

7.3

Makefile Makefile LAB5 := -DLAB5_EX1 -DLAB5_EX2(10)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
LAB3    := -DLAB3_EX1 -DLAB3_EX2
LAB4    := -DLAB4_EX1 -DLAB4_EX2
LAB5    := -DLAB5_EX1 -DLAB5_EX2
# LAB6  := -DLAB6_EX2
# LAB7  := -DLAB7_EX1 #-D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0020000 (phys)
  edata 0xA0153EC0 (phys)
  end   0xA01571A0 (phys)
Kernel executable memory footprint: 1245KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0198000
free pages: 00001E68
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: RR_scheduler
proc_init succeeded
kernel_execve: pid = 2, name = "exit".
I am the parent. Forking the child...
I am parent, fork a child pid 3
I am the parent, waiting now..
I am the child.
waitpid 3 ok.
exit pass.
all user-mode processes have quit.
init check memory pass.
kernel panic at kern/process/proc.c:554:
    initproc exit.

Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
K>
```

7.4

lab5 lab2 3 1 "LAB5" challenge "LAB5" "YOUR CODE" - markdown - - ucore_lab - OS - OS

1:

do_execv load_icode kern/process/proc.c ELF proc_struct trapframe trapframe

CPU ucore CPU RUNNING

2:

do_fork copy_range kern/mm/pmm.c copy_range

"Copy on Write "

Copy-on-write COW A A A " " A " " — B B " " B A

3: FORK/EXEC/WAIT/EXIT

fork/exec/wait/exit - fork/exec/wait/exit - ucore

make qemu -j 16

CHALLENGE COPY ON WRITE COW

, cow

" " ucore

ucore page fault ucore COW

COW bug https://dirtycow.ninja/ ucore COW

big challenge.

7.5

7.5.1 A

ucore CPU

ucore panic kernel_debug_monitor

“ ” “ ” “ ”

3~4 “ ”

“ ” “ ” “ ” CPU “ ” “ ” PCI

“ ”, “ ” “ ” CPU “ ” CPU

-
- CPU CPU “ ”
- “ ”
- “ ” CPU

lab1 CPU CPU CPU

CPU libc libc “ ” “ ” CPU CPU “ ” CPU CPU new ready running blocked exit

new
new
“ ” ready
running 10 swap
CPU running blocked CPU running exit
exit 2 CPU
CPU running ready CPU blocked
blocked ready

8. 6

8.1

8.1.1

-
- ucore Round-Robin
- (Stride Scheduling)

8.1.2

FIFO ucore Round-Robin RR RR Stride Scheduling

8.1.3

FIFO kern/schedule/sched.c schedule FIFO ucore kern/schedule/sched.c default_sched.[ch]

idle cpu scheduler ucore runnable process cpu idle idle schedule ucore (kern/process/proc.c idleproc) cpu idle

init.c kern_init sched_init sched_init sched_class ucore

1.
2.
3.
4.

8.2

8.2.1

ucore runnable ucore struct proc_struct state, running runnable (state) (PROC_RUNNABLE running

- cpu sys_fork uninit (proc.c alloc_proc)
- runnable
- sched_class rq runnable running CPU
- running wait sleeping
- sleeping wakeup runnable
- running exit zombie unused
- runnable

CPU preemptive ucore ucore ucore non-preemptive “ ” CPU ucore “ ”

1. lab7
2. ucore shcedule

CPU CPU CPU “ ” “ ” shedule

schedule

1	proc.c::do_exit	CPU
2	proc.c::do_wait	CPU
3	proc.c::init_main	1. initproc CPU ; 2. initproc kswapd 10
4	proc.c::cpu_idle	idleproc schedule
5	sync.h::lock	CPU
6	trap.c::trap	need_resched 1 CPU

1 2 5 CPU 3 4 initproc schedule idle schedule 6

```
if (!in_kernel) {
    ……

    if (current->need_resched) {
        schedule();
    }
}
```

“ ” need_resched 1 shedule if if ucore racecondition ucore schedule CPU ucore

A trap ( ) A ((1)) A trapframe ucore schedule CPU B proc_run proc_run switch_to B ((2)) B ertn B ((3))

B ((4)) B B trapframe A ucore A((5)) A schedule (switch_to) A A A ((6))

a) cpu

b) forkrets

8.3

8.3.1

CPU …

“ ” “ ” “ ” “ ”

CPU

timer

CPU

- ucore MAX_PROCESS ucore run-queue rq,
- ucore CPU
- list_entry_t, list_entry_t struct proc_struct *, le2proc struct proc_struct run_link list_entry_t run_list struct proc_struct proc = le2proc(, run_link)
- ucore

```
struct sched_class {
    //
    const char *name;
    //
    void (*init) (struct run_queue *rq);
    //     p      rq
    void (*enqueue) (struct run_queue *rq, struct proc_struct *p);
    //     p     rq
    void (*dequeue) (struct run_queue *rq, struct proc_struct *p);
    //
    struct proc_struct* (*pick_next) (struct run_queue *rq);
    // timetick
    void (*proc_tick)(struct  run_queue* rq, struct proc_struct* p);
};
```

- proc.h struct proc_struct

```
struct proc_struct {
    // . . .
    //
    volatile bool need_resched;
    //
    list_entry_t run_link;
    //
    int time_slice;
    // round-robin
    //                 LAB6
    skew_heap_entry_t  lab6_run_pool;
    //              LAB6
    uint32_t lab6_priority;
    //              LAB6
    uint32_t lab6_stride;
};
```

default_sched.c RR ucore RR RR_sched_class

struct run_queue run_queue

```
struct run_queue {
    //
    list_entry_t run_list;
    //               LAB6
    skew_heap_entry_t  *lab6_run_pool;
    //
    unsigned int proc_num;
    //
    int max_time_slice;
};
```

ucore runnable

wakup_proc shedule run_timer_list ucore

wakeup_proc sched_class_enqueue wakeup_proc schedule : CPU “ ” “ ” sched_class_enqueue sched_class_pick_next sched_class_enqueue run_timer_list timer timer sched_class_proc_tick

- sched_class_enqueue
- sched_class_dequeue
- sched_class_pick_next
- sched_class_proc_tick

4 sched_class 4

RR

RR runnable CPU RR runnable RR proc_struct time_slice RR timer time_slice time_slice 0 CPU rq max_time_slice rq

RR_enqueue rq 0 rq max_time_slice

```
static void
RR_enqueue(struct run_queue *rq, struct proc_struct *proc) {
    assert(list_empty(&(proc->run_link)));
    list_add_before(&(rq->run_list), &(proc->run_link));
    if (proc->time_slice == 0 || proc->time_slice > rq->max_time_slice) {
        proc->time_slice = rq->max_time_slice;
    }
    proc->rq = rq;
    rq->proc_num ++;
}
```

RR_pick_next rq

```
static struct proc_struct *
RR_pick_next(struct run_queue *rq) {
    list_entry_t *le = list_next(&(rq->run_list));
    if (le != &(rq->run_list)) {
        return le2proc(le, run_link);
    }
    return NULL;
}
```

RR_dequeue rq proc_num

```
static void
RR_dequeue(struct run_queue *rq, struct proc_struct *proc) {
    assert(!list_empty(&(proc->run_link)) && proc->rq == rq);
    list_del_init(&(proc->run_link));
    rq->proc_num --;
}
```

RR_proc_tick timer trap time_slice time_slice need_resched 1 trap need_resched 1 schedule

```
static void
RR_proc_tick(struct run_queue *rq, struct proc_struct *proc) {
    if (proc->time_slice > 0) {
        proc->time_slice --;
    }
    if (proc->time_slice == 0) {
        proc->need_resched = 1;
    }
}
```

8.4 Stride Scheduling

8.4.1 Stride Scheduling

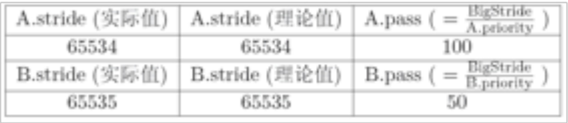
2 **,**

round-robin CPU CPU CPU Stride

- Stride Scheduling
-
- runnable stride pass stride
- runnable stride
- P stride pass
- 2. stride P.pass =BigStride / P.priority P.priority 1 BigStride ucore

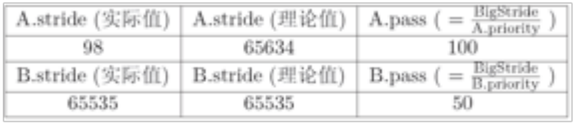
- init:
  -
  - RR
- enqueue
  - proc stride
  - proc
- dequeue
  -
- pick next
  - stride
  - stride pass = BIG_STRIDE / P->priority; P->stride += pass
- proc tick:
  -
  - process rq.max_time_slice

stride stride stride stride A B stride 16



A.stride (实际值)	A.stride (理论值)	A.pass ($= \frac{\text{BigStride}}{\text{A.priority}}$)
65534	65534	100
B.stride (实际值)	B.stride (理论值)	B.pass ($= \frac{\text{BigStride}}{\text{B.priority}}$)
65535	65535	50

A



A.stride (实际值)	A.stride (理论值)	A.pass ($= \frac{\text{BigStride}}{\text{A.priority}}$)
98	65634	100
B.stride (实际值)	B.stride (理论值)	B.pass ($= \frac{\text{BigStride}}{\text{B.priority}}$)
65535	65535	50

stride PASS_MAX Stride STRIDE_MAX STRIDE_MIN

STRIDE_MAX – STRIDE_MIN <= PASS_MAX

1

Priority > 1 STRIDE_MAX – STRIDE_MIN <= BIG_STRIDE, BigStride Stride 0 Stride 98 < 65535 98 - 65535 16 99, 98 > 65535 Stride Stride

2 ucore Stride 32 BigStride

Stride Scheduling

pick_next stride Stride priority Round-Robin Stride

Stride

libs/skew_heap.h

```
//
typedef struct skew_heap_entry  skew_heap_entry_t;
//
void skew_heap_init(skew_heap_entry_t *a);
//     b        a
skew_heap_entry_t  *skew_heap_insert(skew_heap_entry_t  *a,
                                     skew_heap_entry_t  *b,
                                     compare_f comp);
//     b        a
    skew_heap_entry_t  *skew_heap_remove(skew_heap_entry_t  *a,
                                         skew_heap_entry_t  *b,
                                         compare_f comp);
```

comp sched_stride.c proc_stride_comp_f stride Stride

- struct run_queue lab6_run_pool NULL
- struct proc_struct lab6_run_pool LAB6 Stride
- init(rq):
  - Initialize rq->run_list
  - Set rq->lab6_run_pool to NULL
  - Set rq->proc_num to 0
- enqueue(rq, proc)
  - Initialize proc->time_slice
  - Insert proc->lab6_run_pool into rq->lab6_run_pool
  - rq->proc_num ++
- dequeue(rq, proc)
  - Remove proc->lab6_run_pool from rq->lab6_run_pool
  - rq->proc_num --
- pick_next(rq)
  - If rq->lab6_run_pool == NULL, return NULL
  - Find the proc corresponding to the pointer rq->lab6_run_pool
  - proc->lab6_stride += BIG_STRIDE / proc->lab6_priority
  - Return proc
- proc_tick(rq, proc):
  - If proc->time_slice > 0, proc->time_slice --
  - If proc->time_slice == 0, set the flag proc->need_resched

8.5

Makefile Makefile LAB6 := -DLAB6_EX2(11)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
LAB3    := -DLAB3_EX1 -DLAB3_EX2
LAB4    := -DLAB4_EX1 -DLAB4_EX2
LAB5    := -DLAB5_EX1 -DLAB5_EX2
LAB6    := -DLAB6_EX2
# LAB7  := -DLAB7_EX1 #-D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0020000 (phys)
  edata 0xA0153EC0 (phys)
  end   0xA01571A0 (phys)
Kernel executable memory footprint: 1245KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0198000
free pages: 00001E68
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: RR_scheduler
proc_init succeeded
kernel_execve: pid = 2, name = "exit".
I am the parent. Forking the child...
I am parent, fork a child pid 3
I am the parent, waiting now..
I am the child.
waitpid 3 ok.
exit pass.
all user-mode processes have quit.
init check memory pass.
kernel panic at kern/process/proc.c:554:
    initproc exit.

Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
K>
```

8.6

lab6 lab2 2 2 “LAB6” challenge “LAB6” “YOUR CODE” - markdown - - ucore_lab - OS - OS

1: ROUND ROBIN

- sched_class Round Robin ucore - ” “

2: STRIDE SCHEDULING

RR default_sched_stride_c default_sched.c Stride Stride

Stride Stride

- strid-shed paper location1
- strid-shed paper location2
- GOOGLE “Stride Scheduling”

make qemu -j 16

CHALLENGE 1 LINUX CFS

ucore Linux CFS Linux CFS ucore

CHALLENGE 2 UCORE (FIFO, SJF,...)

9. 7

9.1

9.1.1

-
-
- ucore semaphore
- ucore monitor condition variable
-

9.1.2

ucore — semaphore

kern/sync/check_sync.c

9.1.3

ucore wait_queue RUNNABLE STATE CPU

ucore init_main init_main check_sync kern/sync/check_sync.c check_sync

check_sync 5 5 5 5

check_sync 5 5

9.2

9.2.1

- - / wait_queue test_and_set_bit /

. ucore timer timer splice

- sched.h, sched.c timer timer :
- typedef struct {……} timer_t: timer_t sched.h timer_init
- void timer_init(timer t *timer, struct proc_struct *proc, int expires): expires proc
- void add_timer(timer t *timer): timer_t runnable
- void del_timer(timer_t *time):
- void run_timer_list(void): ucore

timer_t

1. timer_t add_timer
2. run_timer_list timer_t
3. run_timer_list timer_t

ucore trap_dispatch ucore

- -

ucore / kern/sync.c local_intr_save(x) local_intr_restore(x) kern/driver intr_enable() intr_disable()

```
    local_intr_save --> __intr_save --> intr_disable --> __lcsr_csrxchg(LISA_CSR_CRMD_IE, LISA_CSR_CRMD_IE, LISA_CSR_CRMD)
    local_intr_restore--> __intr_restore --> intr_enable --> __lcsr_csrxchg(0, LISA_CSR_CRMD_IE, LISA_CSR_CRMD)
```

LoongArch32 CSR.CRMD IE

```
local_intr_save(intr_flag);
{

}
local_intr_restore(intr_flag);
……
```

ucore CPU CPU CPU

, ,

wait_queue

PROC_RUNNABLE ucore kern/sync/{ wait.h, wait.c } wait wait queue ucore wait queue PROC_SLEEPING

```
typedef  struct {
    struct proc_struct *proc;     //
    uint32_t wakeup_flags;        //
    wait_queue_t *wait_queue;     //   wait      wait_queue
    list_entry_t wait_link;       //    wait_queue wait
} wait_t;

typedef struct {
    list_entry_t wait_head;       //wait_queue
} wait_queue_t;

le2wait(le, member)               //  wait_t       wait_t
```

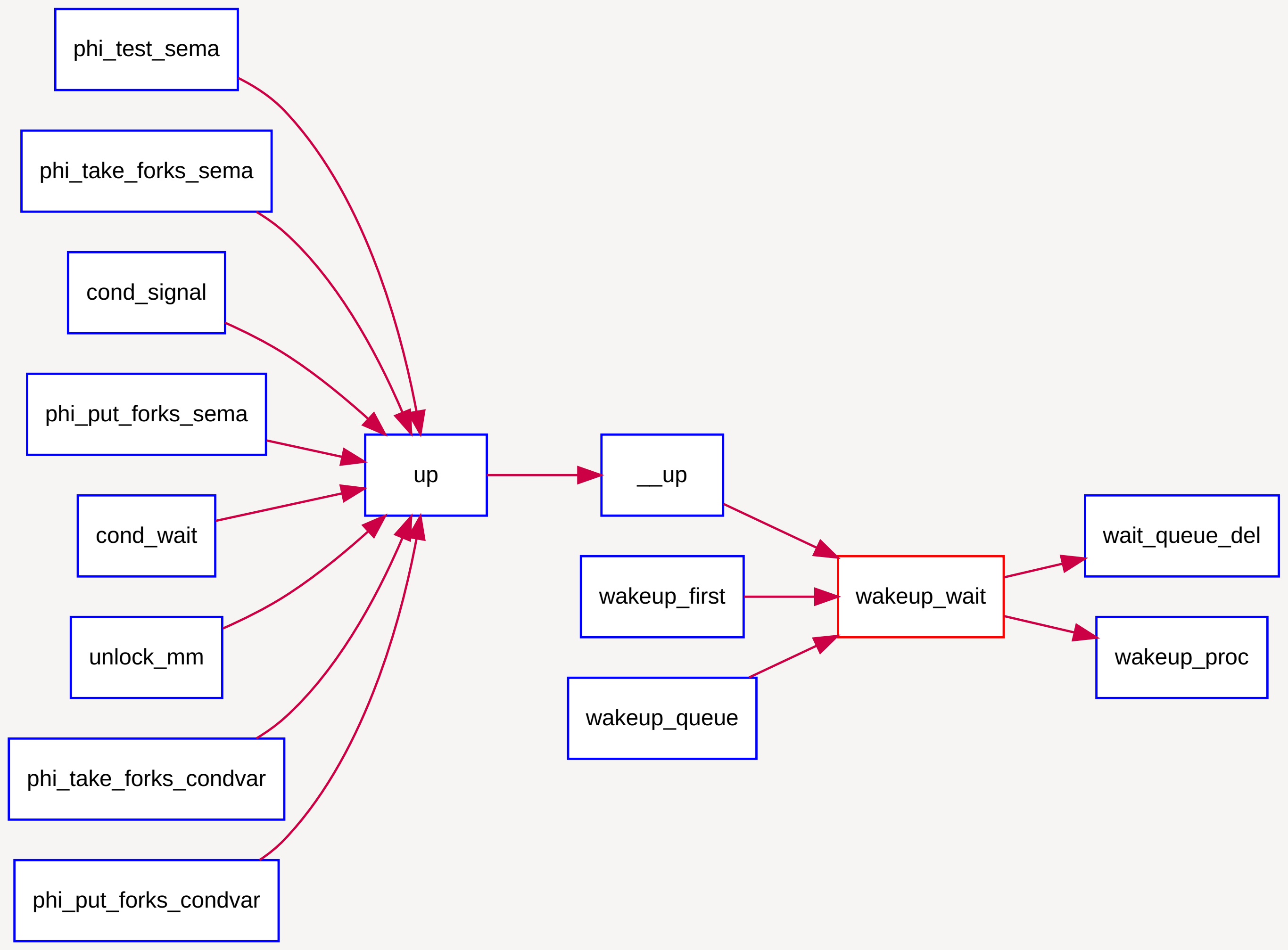
wait wait queue wait queue

```
void wait_init(wait_t *wait, struct proc_struct *proc);    //   wait
bool wait_in_queue(wait_t *wait);                          //wait   wait queue
void wait_queue_init(wait_queue_t *queue);                 //   wait_queue
void wait_queue_add(wait_queue_t *queue, wait_t *wait);    // wait   wait queue
void wait_queue_del(wait_queue_t *queue, wait_t *wait);    // wait queue   wait
wait_t *wait_queue_next(wait_queue_t *queue, wait_t *wait);//  wait
wait_t *wait_queue_prev(wait_queue_t *queue, wait_t *wait);//  wait
wait_t *wait_queue_first(wait_queue_t *queue);             //  wait queue    wait
wait_t *wait_queue_last(wait_queue_t *queue);              //  wait queue     wait
bool wait_queue_empty(wait_queue_t *queue);                //wait queue
```

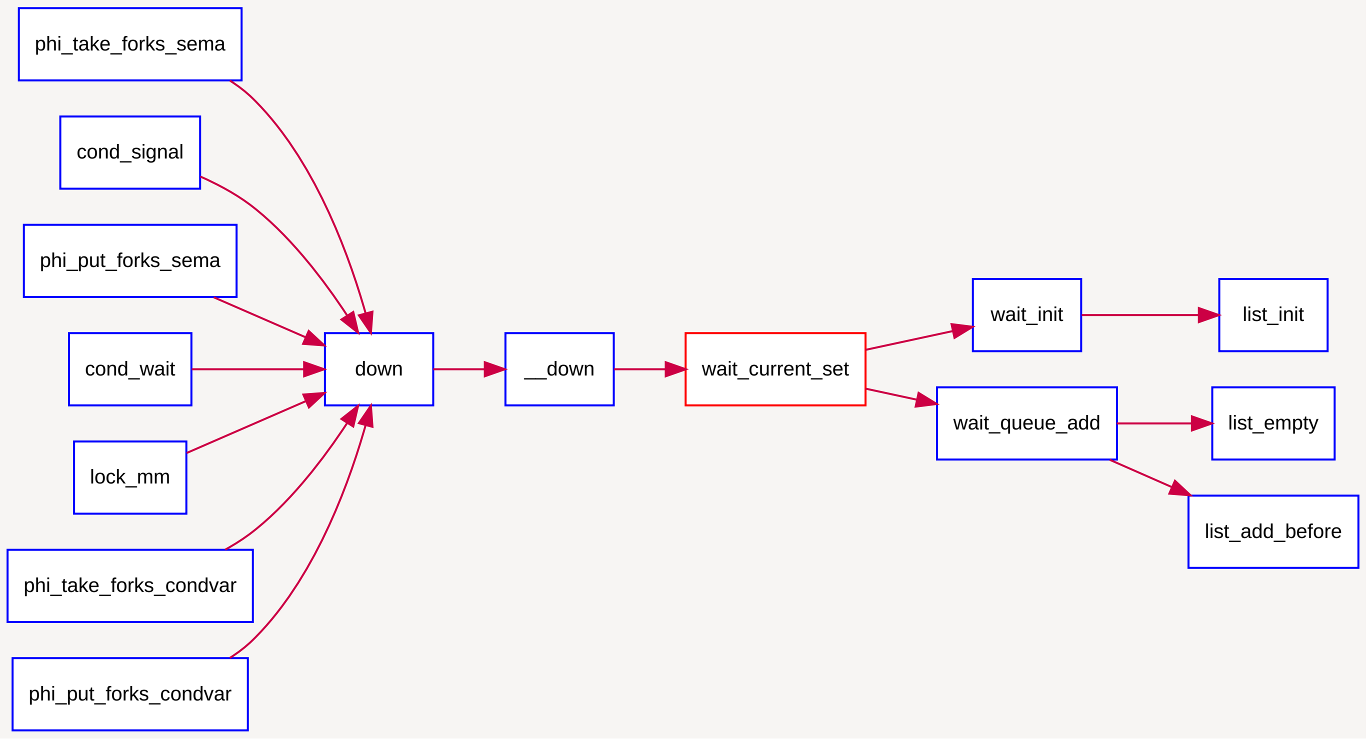
- `wait_current_set` -- `wakeup_wait`

```
// wait              wait      queue
void wait_current_set(wait_queue_t *queue, wait_t *wait, uint32_t wait_state);
//         wait     queue
wait_current_del(queue, wait);
//   wait
void wakeup_wait(wait_queue_t *queue, wait_t *wait, uint32_t wakeup_flags, bool del);
//             wait
void wakeup_first(wait_queue_t *queue, uint32_t wakeup_flags, bool del);
//
void wakeup_queue(wait_queue_t *queue, uint32_t wakeup_flags, bool del);
```

`wakeup_wait` V `up` `wait_queue_del` `wakup_proc`



`wait_current_set` P dowr wait_init wait_queue_add`


phi_take_forks_sema
cond_signal
phi_put_forks_sema
cond_wait
lock_mm
phi_take_forks_condvar
phi_put_forks_condvar
down
__down
wait_current_set
wait_init
list_init
wait_queue_add
list_empty
list_add_before

9.3

spinlock CPU “Operating Systems Internals and Design Principles” “ ”

```
struct semaphore {
int count;
queueType queue;
};
void semWait(semaphore s)
{
s.count--;
if (s.count < 0) {
/* place this process in s.queue */;
/* block this process */;
}
}
void semSignal(semaphore s)
{
s.count++;
if (s.count<= 0) {
/* remove a process P from s.queue */;
/* place process P on ready list */;
}
}
```

>1 s V semSignal(s) s P semWait(s)

ucore wait_queue

```
typedef struct {
    int value;                   //
    wait_queue_t wait_queue;     //
} semaphore_t;
```

semaphore_t record semaphore) value wait_queue

ucore P down(semaphore_t *sem) V up(semaphore_t *sem) __down(semaphore_t *sem, uint32_t wait_state) __up(semaphore_t *sem, uint32_t wait_state)

● __down(semaphore_t *sem, uint32_t wait_state, timer_t *timer) P value 0 >0 value >0 V wait

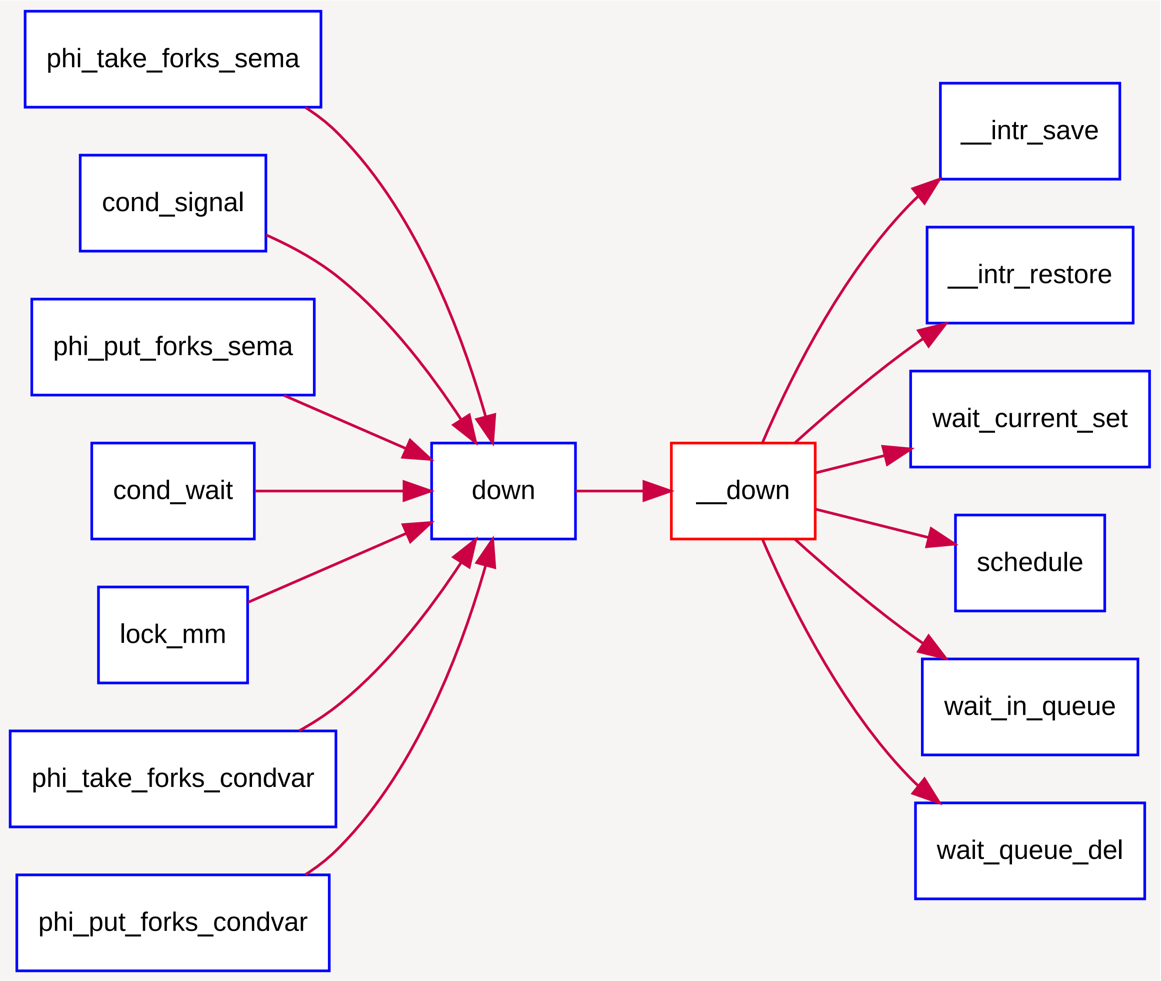
```
static __noinline uint32_t __down(semaphore_t *sem, uint32_t wait_state) {
    bool intr_flag;
    local_intr_save(intr_flag);
    if (sem->value > 0) {
        sem->value --;
        local_intr_restore(intr_flag);
        return 0;
    }
    wait_t __wait, *wait = &__wait;
    wait_current_set(&(sem->wait_queue), wait, wait_state);
    local_intr_restore(intr_flag);

    schedule();

    local_intr_save(intr_flag);
    wait_current_del(&(sem->wait_queue), wait);
    local_intr_restore(intr_flag);

    if (wait->wakeup_flags != wait_state) {
        return wait->wakeup_flags;
    }
    return 0;
}
```

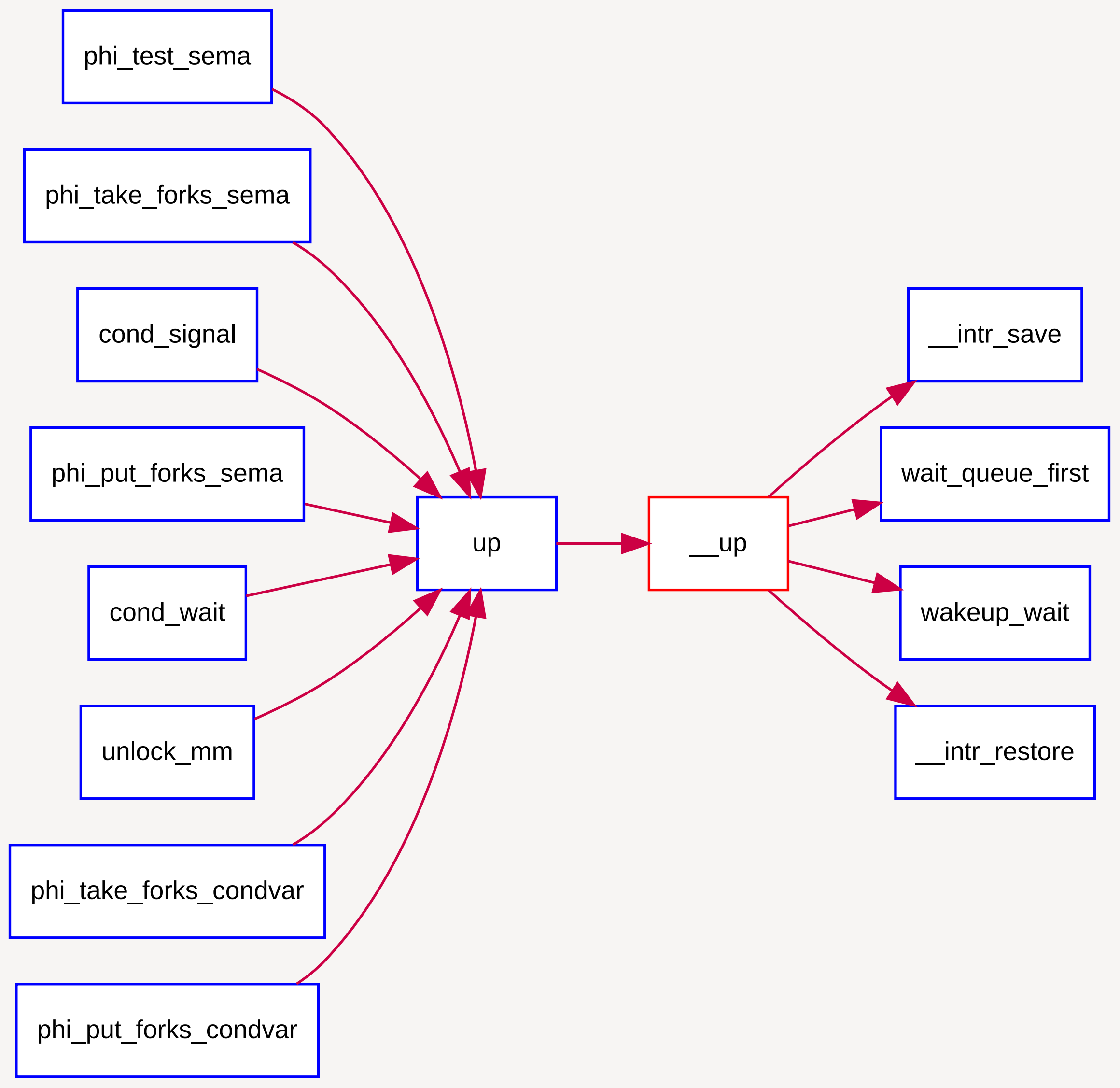
`__down`



- __up(semaphore_t *sem, uint32_t wait_state) V wait queue value semophore wakeup_wait waitqueue wait wait

```
static __noinline void __up(semaphore_t *sem, uint32_t wait_state) {
    bool intr_flag;
    local_intr_save(intr_flag);
    {
        wait_t *wait;
        if ((wait = wait_queue_first(&(sem->wait_queue))) == NULL) {
            sem->value ++;
        }
        else {
            wakeup_wait(&(sem->wait_queue), wait, wait_state, 1);
        }
    }
    local_intr_restore(intr_flag);
}
```

`__up`



value

- value>0
- vlaue<0
- value=0

9.4

Hansan “ ”

-
-
-
-

Cond (busy waiting)

```
while not( Cond ) do {}
```

Cond Condition Variables CV CV Cond Cond Cond CV Cond Cond CV CV

- wait_cv Pc .
- signal_cv Pc Pc

" "

OS Concept 6.7.2 “ ”

```
monitor dp
{
    enum {THINKING, HUNGRY, EATING} state[5];
    condition self[5];

    void pickup(int i) {
        state[i] = HUNGRY;
        test(i);
        if (state[i] != EATING)
            self[i].wait_cv();
    }

    void putdown(int i) {
        state[i] = THINKING;
        test((i + 4) % 5);
        test((i + 1) % 5);
    }

    void test(int i) {
        if ((state[(i + 4) % 5] != EATING) &&
           (state[i] == HUNGRY) &&
           (state[(i + 1) % 5] != EATING)) {
              state[i] = EATING;
              self[i].signal_cv();
        }
    }

    initialization code() {
        for (int i = 0; i < 5; i++)
        state[i] = THINKING;
        }
}
```

java C OS ucore C

ucore ucore monitor_t

```c
typedef struct monitor{
    semaphore_t mutex;      // the mutex lock for going into the routines in monitor, should be initialized to 1
    // the next semaphore is used to
    //    (1) procs which call cond_signal funciton should DOWN next sema after UP cv.sema
    // OR (2) procs which call cond_wait funciton should UP next sema before DOWN cv.sema
    semaphore_t next;
    int next_count;         // the number of of sleeped procs which cond_signal funciton
    condvar_t *cv;          // the condvars in monitor
} monitor_t;
```

mutex cv `wait_cv` Cond Cond `signal_cv` Cond

next next_count cv `signal_cv` A `wait_cv` B B B A B A next next_count `singal_cv`

condvar_t

```c
typedef struct condvar{
    semaphore_t sem;    // the sem semaphore is used to down the waiting proc, and the signaling proc should up the waiting proc
    int count;           // the number of waiters on condvar
    monitor_t * owner;  // the owner(monitor) of this condvar
} condvar_t;
```

sem `wait_cv` Cond `signal_cv` sem count owner

SIGNAL WAIT

ucore `wait_cv` `signal_cv` `cond_wait` `cond_signal` `cond_init` `cond_wait(condvar_t *cvp, semaphore_t *mp)` `cond_signal (condvar_t *cvp)` OS Concept 6.7.3 “ ” `wait_cv`

wait_cv

```c
cv.count++;
if(monitor.next_count > 0)
   sem_signal(monitor.next);
else
   sem_signal(monitor.mutex);
sem_wait(cv.sem);
cv.count -- ;
```

`cond_wait` A `cond_wait` Cond cv.count

monitor.next_count 0 1 cond_signal monitor.next monitor.next B A cv.sem A cv.count

A `cond_signal` B `cond_wait` A B

: cond_wait sem_signal(mutex) sem_wait(mutex) wait(mutex)

monitor.next_count 0 cond_signal monitor.mutex A cv.sem cv.count

`signal_cv`

** signal_cv **

```c
if( cv.count > 0) {
   monitor.next_count ++;
   sem_signal(cv.sem);
   sem_wait(monitor.next);
   monitor.next_count -- ;
}
```

`cond_signal` B cv.count 0 cond_wait 0 cond_wait A cv.sem A B A monitor.next_count B monitor.next monitor.next_count

```
function_in_monitor  …
{
  sem.wait(monitor.mutex);
//-----------------------------
  the real body of function;
//-----------------------------
  if(monitor.next_count > 0)
     sem_signal(monitor.next);
  else
     sem_signal(monitor.mutex);
}
```

1 2 cond_signal A cond_signal monitor.next_count 0 sem_wait(monitor.next) monitor.next_count 0 0 sem_signal(monitor.next) cond_signal

9.5

Makefile Makefile LAB7 := -DLAB7_EX1 -D_SHOW_PHI(12)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
LAB3    := -DLAB3_EX1 -DLAB3_EX2
LAB4    := -DLAB4_EX1 -DLAB4_EX2
LAB5    := -DLAB5_EX1 -DLAB5_EX2
LAB6    := -DLAB6_EX2
LAB7    := -DLAB7_EX1 -D_SHOW_PHI
# LAB8  := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0021000 (phys)
  edata 0xA0155490 (phys)
  end   0xA0158770 (phys)
Kernel executable memory footprint: 1246KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0199000
free pages: 00001E67
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: stride_scheduler
proc_init succeeded
kernel_execve: pid = 2, name = "exit".
I am the parent. Forking the child...
I am parent, fork a child pid 13
I am the parent, waiting now..
I am No.0 philosopher_sema
Iter 1, No.0 philosopher_sema is thinking
I am No.1 philosopher_sema
Iter 1, No.1 philosopher_sema is thinking
I am No.2 philosopher_sema
Iter 1, No.2 philosopher_sema is thinking
I am No.2 philosopher_condvar
Iter 1, No.2 philosopher_condvar is thinking
I am No.3 philosopher_sema
Iter 1, No.3 philosopher_sema is thinking
I am No.4 philosopher_sema
Iter 1, No.4 philosopher_sema is thinking
I am No.0 philosopher_condvar
Iter 1, No.0 philosopher_condvar is thinking
I am No.1 philosopher_condvar
Iter 1, No.1 philosopher_condvar is thinking
I am No.3 philosopher_condvar
Iter 1, No.3 philosopher_condvar is thinking
I am No.4 philosopher_condvar
Iter 1, No.4 philosopher_condvar is thinking
I am the child.
waitpid 13 ok.
exit pass.
Iter 1, No.0 philosopher_sema is eating
Iter 1, No.2 philosopher_sema is eating
phi_test_condvar: state_condvar[2] will eating
phi_test_condvar: signal self_cv[2]
cond_signal begin: cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
cond_signal end: cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
Iter 1, No.2 philosopher_condvar is eating
phi_test_condvar: state_condvar[0] will eating
```

```
phi_test_condvar: signal self_cv[0]
cond_signal begin: cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
cond_signal end: cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
Iter 1, No.0 philosopher_condvar is eating
phi_take_forks_condvar: 1 didn't get fork and will wait
cond_wait begin:  cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
phi_take_forks_condvar: 3 didn't get fork and will wait
cond_wait begin:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
phi_take_forks_condvar: 4 didn't get fork and will wait
cond_wait begin:  cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.0 philosopher_sema is thinking
Iter 2, No.2 philosopher_sema is thinking
Iter 1, No.1 philosopher_sema is eating
phi_test_condvar: state_condvar[3] will eating
phi_test_condvar: signal self_cv[3]
cond_signal begin: cvp a01b20cc, cvp->count 1, cvp->owner->next_count 0
Iter 1, No.4 philosopher_sema is eating
cond_wait end:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 1
Iter 1, No.3 philosopher_condvar is eating
cond_signal end: cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.2 philosopher_condvar is thinking
phi_test_condvar: state_condvar[1] will eating
phi_test_condvar: signal self_cv[1]
cond_signal begin: cvp a01b20a4, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20a4, cvp->count 0, cvp->owner->next_count 1
Iter 1, No.1 philosopher_condvar is eating
cond_signal end: cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.0 philosopher_condvar is thinking
phi_take_forks_condvar: 0 didn't get fork and will wait
cond_wait begin:  cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[0] will eating
phi_test_condvar: signal self_cv[0]
cond_signal begin: cvp a01b2090, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b2090, cvp->count 0, cvp->owner->next_count 1
Iter 2, No.0 philosopher_condvar is eating
cond_signal end: cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.1 philosopher_condvar is thinking
Iter 2, No.1 philosopher_sema is thinking
phi_take_forks_condvar: 2 didn't get fork and will wait
cond_wait begin:  cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.2 philosopher_sema is eating
Iter 2, No.4 philosopher_sema is thinking
phi_test_condvar: state_condvar[2] will eating
phi_test_condvar: signal self_cv[2]
cond_signal begin: cvp a01b20b8, cvp->count 1, cvp->owner->next_count 0
Iter 2, No.0 philosopher_sema is eating
cond_wait end:  cvp a01b20b8, cvp->count 0, cvp->owner->next_count 1
Iter 2, No.2 philosopher_condvar is eating
cond_signal end: cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.3 philosopher_condvar is thinking
Iter 3, No.0 philosopher_sema is thinking
Iter 3, No.2 philosopher_condvar is thinking
phi_test_condvar: state_condvar[4] will eating
phi_test_condvar: signal self_cv[4]
cond_signal begin: cvp a01b20e0, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20e0, cvp->count 0, cvp->owner->next_count 1
Iter 1, No.4 philosopher_condvar is eating
Iter 3, No.2 philosopher_sema is thinking
Iter 1, No.3 philosopher_sema is eating
cond_signal end: cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
Iter 3, No.0 philosopher_condvar is thinking
phi_test_condvar: state_condvar[1] will eating
phi_test_condvar: signal self_cv[1]
cond_signal begin: cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
cond_signal end: cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.1 philosopher_condvar is eating
phi_take_forks_condvar: 3 didn't get fork and will wait
cond_wait begin:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.1 philosopher_sema is eating
Iter 2, No.3 philosopher_sema is thinking
Iter 2, No.4 philosopher_sema is eating
phi_take_forks_condvar: 0 didn't get fork and will wait
cond_wait begin:  cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
Iter 3, No.1 philosopher_condvar is thinking
phi_test_condvar: state_condvar[3] will eating
phi_test_condvar: signal self_cv[3]
cond_signal begin: cvp a01b20cc, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 1
Iter 2, No.3 philosopher_condvar is eating
Iter 3, No.1 philosopher_sema is thinking
Iter 3, No.2 philosopher_sema is eating
cond_signal end: cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[0] will eating
phi_test_condvar: signal self_cv[0]
cond_signal begin: cvp a01b2090, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b2090, cvp->count 0, cvp->owner->next_count 1
Iter 3, No.0 philosopher_condvar is eating
cond_signal end: cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
Iter 2, No.4 philosopher_condvar is thinking
phi_take_forks_condvar: 2 didn't get fork and will wait
cond_wait begin:  cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
phi_take_forks_condvar: 4 didn't get fork and will wait
cond_wait begin:  cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
Iter 4, No.0 philosopher_condvar is thinking
Iter 3, No.4 philosopher_sema is thinking
phi_test_condvar: state_condvar[1] will eating
phi_test_condvar: signal self_cv[1]
cond_signal begin: cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
cond_signal end: cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
Iter 3, No.1 philosopher_condvar is eating
phi_test_condvar: state_condvar[4] will eating
phi_test_condvar: signal self_cv[4]
cond_signal begin: cvp a01b20e0, cvp->count 1, cvp->owner->next_count 0
Iter 3, No.0 philosopher_sema is eating
Iter 4, No.2 philosopher_sema is thinking
Iter 2, No.3 philosopher_sema is eating
cond_wait end:  cvp a01b20e0, cvp->count 0, cvp->owner->next_count 1
Iter 2, No.4 philosopher_condvar is eating
cond_signal end: cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
Iter 3, No.3 philosopher_condvar is thinking
phi_take_forks_condvar: 3 didn't get fork and will wait
cond_wait begin:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[3] will eating
phi_test_condvar: signal self_cv[3]
cond_signal begin: cvp a01b20cc, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 1
Iter 3, No.3 philosopher_condvar is eating
cond_signal end: cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
Iter 3, No.4 philosopher_condvar is thinking
Iter 4, No.0 philosopher_sema is thinking
Iter 3, No.1 philosopher_sema is eating
Iter 3, No.3 philosopher_sema is thinking
Iter 4, No.1 philosopher_condvar is thinking
Iter 3, No.4 philosopher_sema is eating
phi_test_condvar: state_condvar[0] will eating
phi_test_condvar: signal self_cv[0]
cond_signal begin: cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
cond_signal end: cvp a01b2090, cvp->count 0, cvp->owner->next_count 0
Iter 4, No.0 philosopher_condvar is eating
Iter 4, No.4 philosopher_sema is thinking
No.0 philosopher_condvar quit
phi_test_condvar: state_condvar[2] will eating
phi_test_condvar: signal self_cv[2]
cond_signal begin: cvp a01b20b8, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20b8, cvp->count 0, cvp->owner->next_count 1
Iter 3, No.2 philosopher_condvar is eating
Iter 4, No.1 philosopher_sema is thinking
Iter 4, No.2 philosopher_sema is eating
cond_signal end: cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
Iter 4, No.3 philosopher_condvar is thinking
phi_test_condvar: state_condvar[4] will eating
phi_test_condvar: signal self_cv[4]
cond_signal begin: cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
cond_signal end: cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
Iter 3, No.4 philosopher_condvar is eating
Iter 4, No.0 philosopher_sema is eating
phi_take_forks_condvar: 1 didn't get fork and will wait
cond_wait begin:  cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
phi_take_forks_condvar: 3 didn't get fork and will wait
cond_wait begin:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
Iter 4, No.4 philosopher_condvar is thinking
No.0 philosopher_sema quit
phi_test_condvar: state_condvar[1] will eating
phi_test_condvar: signal self_cv[1]
cond_signal begin: cvp a01b20a4, cvp->count 1, cvp->owner->next_count 0
Iter 4, No.4 philosopher_sema is eating
cond_wait end:  cvp a01b20a4, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.1 philosopher_condvar is eating
cond_signal end: cvp a01b20a4, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[3] will eating
phi_test_condvar: signal self_cv[3]
cond_signal begin: cvp a01b20cc, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20cc, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.3 philosopher_condvar is eating
No.2 philosopher_sema quit
Iter 4, No.1 philosopher_sema is eating
cond_signal end: cvp a01b20cc, cvp->count 0, cvp->owner->next_count 0
Iter 4, No.2 philosopher_condvar is thinking
phi_take_forks_condvar: 4 didn't get fork and will wait
cond_wait begin:  cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
No.1 philosopher_sema quit
phi_take_forks_condvar: 2 didn't get fork and will wait
cond_wait begin:  cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
No.4 philosopher_sema quit
Iter 3, No.3 philosopher_sema is eating
No.1 philosopher_condvar quit
phi_test_condvar: state_condvar[2] will eating
phi_test_condvar: signal self_cv[2]
cond_signal begin: cvp a01b20b8, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20b8, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.2 philosopher_condvar is eating
cond_signal end: cvp a01b20b8, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[4] will eating
phi_test_condvar: signal self_cv[4]
cond_signal begin: cvp a01b20e0, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp a01b20e0, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.4 philosopher_condvar is eating
cond_signal end: cvp a01b20e0, cvp->count 0, cvp->owner->next_count 0
```

```
No.3 philosopher_condvar quit
Iter 4, No.3 philosopher_sema is thinking
No.4 philosopher_condvar quit
No.2 philosopher_condvar quit
Iter 4, No.3 philosopher_sema is eating
No.3 philosopher_sema quit
all user-mode processes have quit.
init check memory pass.
kernel panic at kern/process/proc.c:554:
    initproc exit.

Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
K>
```

9.6

lab7 lab2 2 2 “LAB7” challenge “LAB7” “YOUR CODE” - markdown - - ucore_lab - OS - OS

1:

/

2:

make qemu -j 16

/

CHALLENGE UCORE

ucore / monitor

CHALLENGE LINUX RCU UCORE RCU

ucore Linux RCU Linux RCU ucore

- http://www.ibm.com/developerworks/cn/linux/l-rcu/
- http://www.diybl.com/course/6_system/linux/Linuxjs/20081117/151814.html

10. 8

10.1

10.1.1

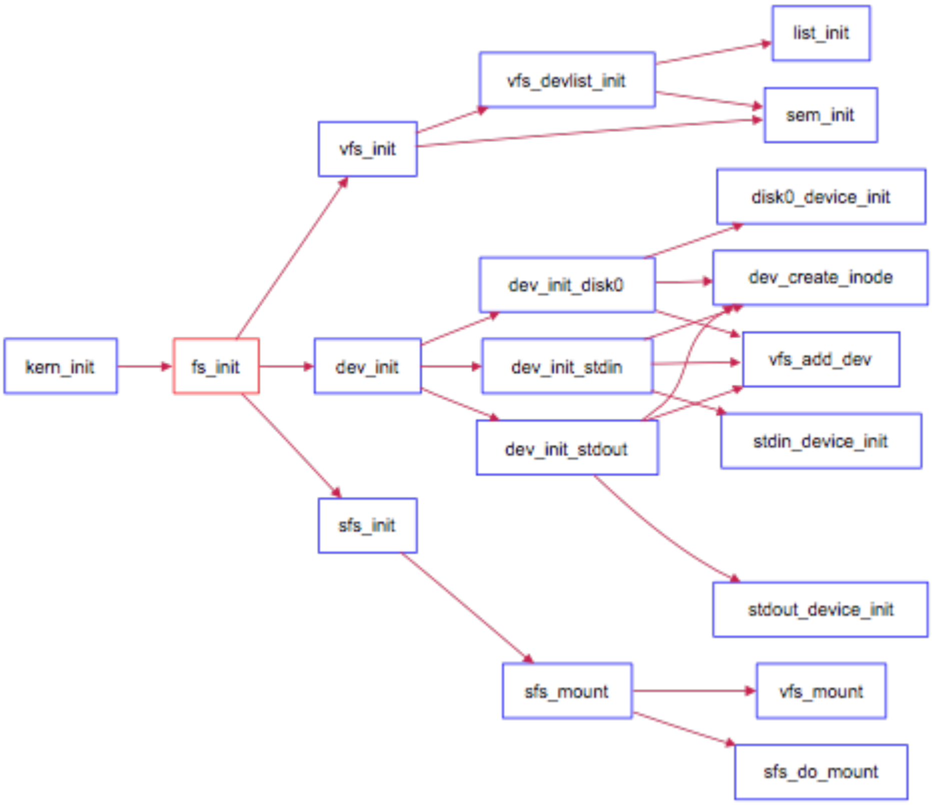
-
- Simple FS
- -VFS

10.1.2

ucore do_execve

ucore

kern_init lab7 fs_init fs_init vfs_init dev_init Simple FS sfs_init SFS



vfs_init device list vdev_list dev_init disk0/ stdin/stdout_device_init inode vdev_list sfs_init Simple FS ucore SFS

10.2

10.2.1 ucore

ucore Havard OS161 Linux UNIX UNIX (file) (dentry) (inode) (mount point)

- UNIX buffer
- UNIX "/test/testfile" "/" "test" "testfile"
- UNIX
- UNIX

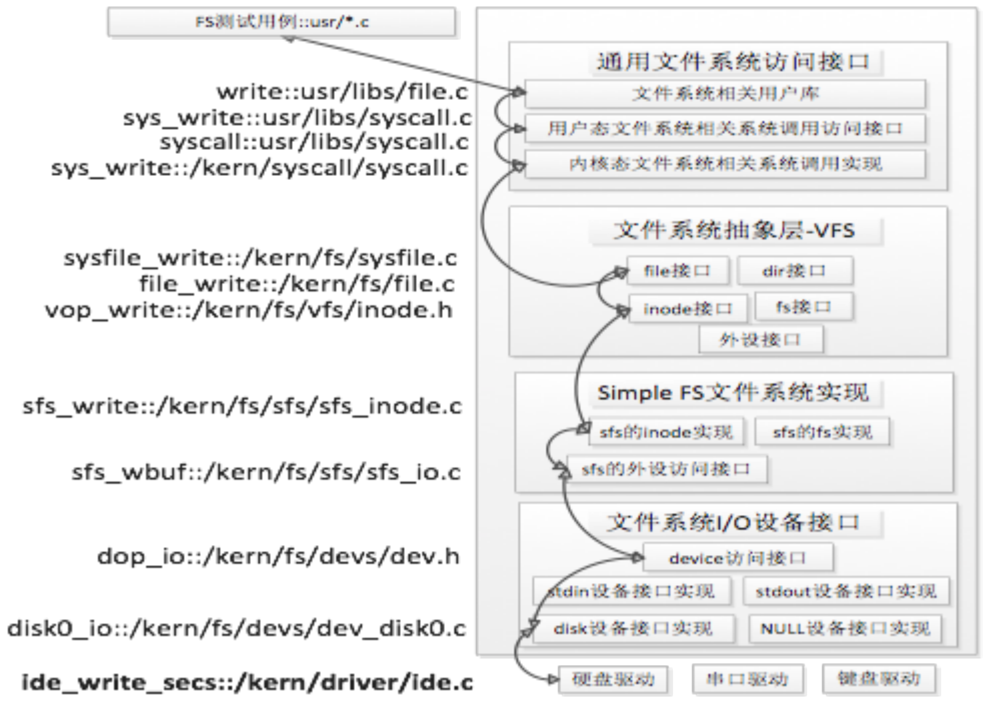
UNIX

ucore UNIX ucore

- ucore
-
- Simple FS
- device disk / /

/ / /

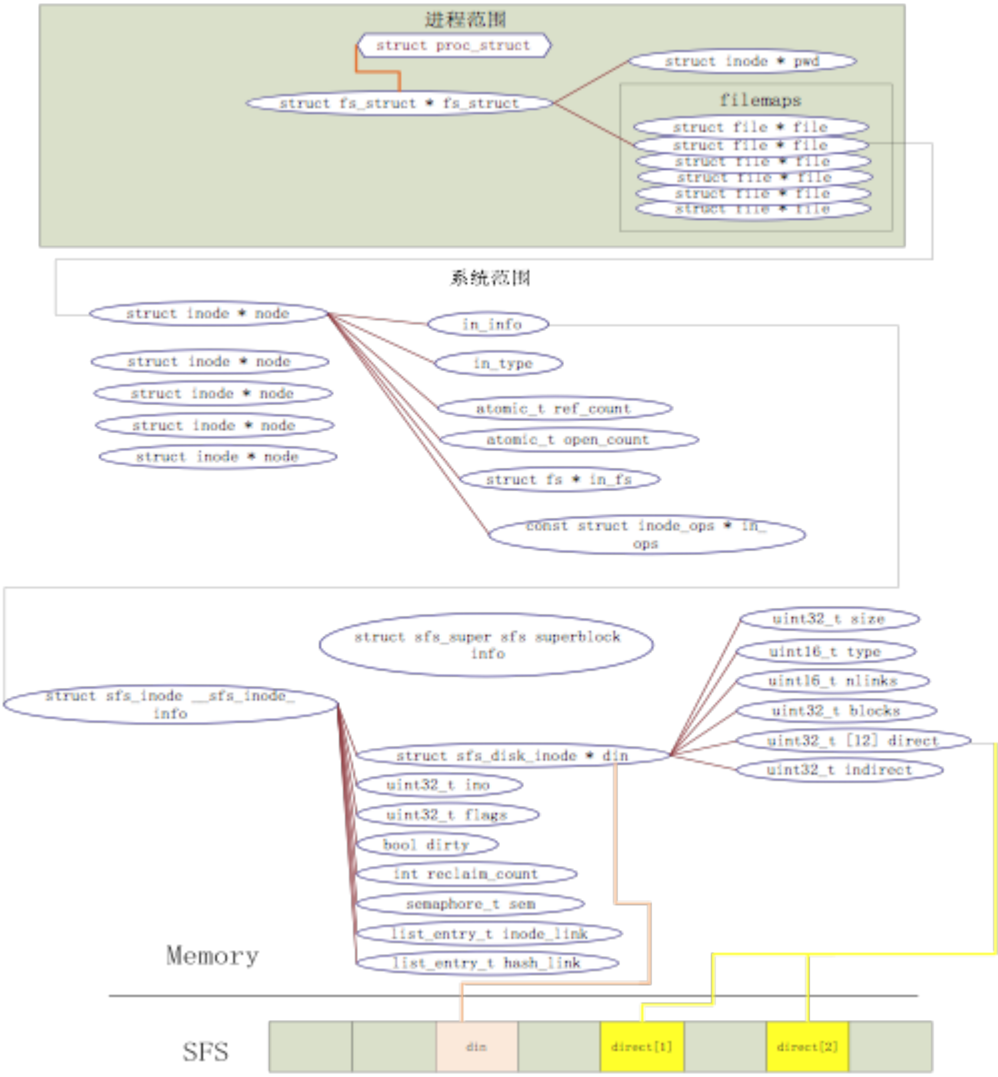
SFS Simple FS block write ucore


FS测试用例::usr/*.c
write::usr/libs/file.c
sys_write::usr/libs/syscall.c
syscall::usr/libs/syscall.c
sys_write::/kern/syscall/syscall.c
sysfile_write::/kern/fs/sysfile.c
file_write::/kern/fs/file.c
vop_write::/kern/fs/vfs/inode.h
sfs_write::/kern/fs/sfs/sfs_inode.c
sfs_wbuf::/kern/fs/sfs/sfs_io.c
dop_io::/kern/fs/devs/dev.h
disk0_io::/kern/fs/devs/dev_disk0.c
ide_write_secs::/kern/driver/ide.c
通用文件系统访问接口
文件系统相关用户库
用户态文件系统相关系统调用访问接口
内核态文件系统相关系统调用实现
文件系统抽象层-VFS
file接口
dir接口
inode接口
fs接口
外设接口
Simple FS文件系统实现
sfs的inode实现
sfs的fs实现
sfs的外设访问接口
文件系统I/O设备接口
device访问接口
stdin设备接口实现
stdout设备接口实现
disk设备接口实现
NULL设备接口实现
硬盘驱动
串口驱动
键盘驱动

ucore ucore ,

- SuperBlock OS
- inode OS
- dentry / OS SFS inode(struct sfs_disk_inode) dentry struct sfs_disk_entry ino inode number file name( ).
- file

ucore



Lab8

open close read write open open O_RDONLY O_WRONLY O_RDWR fd close fd

read write read C

```
count = read(filehandle, buffer, nbytes);
```

count nbytes

count -1 write

chdir opendir readdir closedir ucore opendir closedir open close readdir sys_getdirentry

open close read write sys_open sys_close sys_read sys_write readdir sys_getdirentry syscall ucore ucore file dir

10.3 VFS

10.3.1 - VFS

file & dir

file&dir file

```
struct file {
    enum {
        FD_NONE, FD_INIT, FD_OPENED, FD_CLOSED,
    } status;                         //
    bool readable;                    //
    bool writable;                    //
    int fd;                           //   filemap
    off_t pos;                        //
    struct inode *node;               //        inode
    int open_count;                   //
};
```

kern/process/proc.h proc_struct files_struct

```
struct files_struct {
    struct inode *pwd;                //           inode
    struct file *fd_array;            //
    atomic_t files_count;             //
    semaphore_t files_sem;            //         fs_struct
};
```

files_struct files_struct fd_array file file node inode

inode

index node VFS

```
struct inode {
    union {                                 //          inode   union
        struct device __device_info;          //        inode
        struct sfs_inode __sfs_inode_info;    //SFS      inode
    } in_info;
    enum {
        inode_type_device_info = 0x1234,
        inode_type_sfs_inode_info,
    } in_type;                          // inode
    atomic_t ref_count;                 // inode
    atomic_t open_count;                //   inode
    struct fs *in_fs;                   //
    const struct inode_ops *in_ops;     //   inode       inode
};
```

inode in_ops inode

```
struct inode_ops {
    unsigned long vop_magic;
    int (*vop_open)(struct inode *node, uint32_t open_flags);
    int (*vop_close)(struct inode *node);
    int (*vop_read)(struct inode *node, struct iobuf *iob);
    int (*vop_write)(struct inode *node, struct iobuf *iob);
    int (*vop_getdirentry)(struct inode *node, struct iobuf *iob);
    int (*vop_create)(struct inode *node, const char *name, bool excl, struct inode **node_store);
int (*vop_lookup)(struct inode *node, char *path, struct inode **node_store);
……
 };
```

SFS inode_ops

10.4

10.4.1 IO

stdin stdout disk0 stdin stdout CONSOLE disk0 SFS ucore

ucore struct device

```
struct device {
    size_t d_blocks;    //
    size_t d_blocksize;  //
    int (*d_open)(struct device *dev, uint32_t open_flags);  //
    int (*d_close)(struct device *dev); //
    int (*d_io)(struct device *dev, struct iobuf *iob, bool write); //
    int (*d_ioctl)(struct device *dev, int op, void *data); // ioctl
};
```

ucore vdev_list ucore

inode device inode vfs_dev_t

```
// device info entry in vdev_list
typedef struct {
    const char *devname;
    struct inode *devnode;
    struct fs *fs;
    bool mountable;
    list_entry_t vdev_link;
} vfs_dev_t;
```

vfs_dev_t vfs_dev_t device inode inode in_type 0x1234 inode in_info device inode inode

stdout

stdout inode

```
kern_init-->fs_init-->dev_init-->dev_init_stdout --> dev_create_inode
                 --> stdout_device_init
                 --> vfs_add_dev
```

dev_init_stdout stdout inode stdout_device_init inode inode->__device_info

stdout console CGA stdout

stdout stdout_device_init

```
static void
stdout_device_init(struct device *dev) {
    dev->d_blocks = 0;
    dev->d_blocksize = 1;
    dev->d_open = stdout_open;
    dev->d_close = stdout_close;
    dev->d_io = stdout_io;
    dev->d_ioctl = stdout_ioctl;
}
```

stdout_open open flags O_WRONLY

stdout_io

```
static int
stdout_io(struct device *dev, struct iobuf *iob, bool write) {
    if (write) {
        char *data = iob->io_base;
        for (; iob->io_resid != 0; iob->io_resid --) {
            kputchar(*data ++);
        }
        return 0;
    }
    return -E_INVAL;
}
```

iob->io_base iob->io_resid 0 cputchar console CGA stdout_io **-**E_INVAL

stdin

stdin stdin

stdin stdin_device_init

```
static void
stdin_device_init(struct device *dev) {
    dev->d_blocks = 0;
    dev->d_blocksize = 1;
    dev->d_open = stdin_open;
    dev->d_close = stdin_close;
    dev->d_io = stdin_io;
    dev->d_ioctl = stdin_ioctl;

    p_rpos = p_wpos = 0;
    wait_queue_init(wait_queue);
}
```

stdout stdin stdin_buffer p_rpos p_wpos wait_queue stdin_device_init p_rpos p_wpos wait_queue

stdin_io

```
static int
stdin_io(struct device *dev, struct iobuf *iob, bool write) {
    if (!write) {
        int ret;
        if ((ret = dev_stdin_read(iob->io_base, iob->io_resid)) > 0) {
            iob->io_resid -= ret;
        }
        return ret;
    }
    return -E_INVAL;
}
```

stdin_io dev_stdin_read dev_stdin_read

```
static int
dev_stdin_read(char *buf, size_t len) {
    int ret = 0;
    bool intr_flag;
    local_intr_save(intr_flag);
    {
        for (; ret < len; ret ++, p_rpos ++) {
        try_again:
            if (p_rpos < p_wpos) {
                *buf ++ = stdin_buffer[p_rpos % stdin_BUFSIZE];
            }
            else {
                wait_t __wait, *wait = &__wait;
                wait_current_set(wait_queue, wait, WT_KBD);
                local_intr_restore(intr_flag);

                schedule();

                local_intr_save(intr_flag);
                wait_current_del(wait_queue, wait);
                if (wait->wakeup_flags == WT_KBD) {
                    goto try_again;
                }
                break;
            }
        }
    }
    local_intr_restore(intr_flag);
    return ret;
}
```

p_rpos < p_wpos stdin_buffer stdin_buffer iobuf p_rpos >=p_wpos read

lab1 QEMU trap_dispatch dev_stdin_write stdin_buffer

10.5 Simple File System

10.5.1 Simple FS

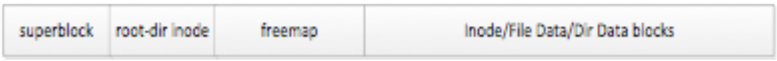
Simple FS SFS ucore ucore ucore

- SFS
- entry entry index node
-
-
-

lab8 SFS hardlink SFS SFS

ucore initrd ucore disk0 SFS Simple Filesystem Sector SFS block 4K page

SFS



superblock	root-dir inode	freemap	Inode/File Data/Dir Data blocks

0 4K superblock

```
struct sfs_super {
    uint32_t magic;                                  /* magic number, should be SFS_MAGIC */
    uint32_t blocks;                                 /* # of blocks in fs */
    uint32_t unused_blocks;                         /* # of unused blocks in fs */
    char info[SFS_MAX_INFO_LEN + 1];                /* infomation for sfs  */
};
```

magic 0x2f8dbe2a SFS img blocks SFS block img unused_block SFS block info "simple file system"

1 root-dir inode inode root-dir SFS root-dir inode

2 SFS 1 bit SFS freemap freemap kern/fs/sfs/bitmap.[ch] bit

inode inode 4096B inode block

sfs_fs.c sfs_do_mount SFS superblock freemap SFS

SFS sfs_disk_inode sfs_disk_entry inode

SFS

```
struct sfs_disk_inode {
    uint32_t size;                              inode        size
    uint16_t type;                                inode
    uint16_t nlinks;                              inode
    uint32_t blocks;                             inode
    uint32_t direct[SFS_NDIRECT];               inode           SFS_NDIRECT
    uint32_t indirect;                             inode
};
```

inode direct[] indirect indirect indirect block

ucore SFS_NDIRECT 12 12 * 4k = 48k ucore 12 * 4k + 1024 * 4k = 48k + 4m 0 inode blocks block indiret 0 block 0 super block

block

```
/* file entry (on disk) */
struct sfs_disk_entry {
    uint32_t ino;
    char name[SFS_MAX_FNAME_LEN + 1];
};
```

inode inode SFS inode block inode root block inode 1 sfs_disk_entry name ino block block inode ino 0 entry

inode sfs_dirent_entry block

```
/* inode for sfs */
struct sfs_inode {
    struct sfs_disk_inode *din;                     /* on-disk inode */
    uint32_t ino;                                   /* inode number */
    uint32_t flags;                                 /* inode flags */
    bool dirty;                                     /* true if inode modified */
    int reclaim_count;                              /* kill inode if it hits zero */
    semaphore_t sem;                                /* semaphore for din */
    list_entry_t inode_link;                        /* entry for linked-list in sfs_fs */
    list_entry_t hash_link;                         /* entry for hash linked-list in sfs_fs */
};
```

SFS inode SFS inode inode inode

entry inode SFS

1. sfs_bmap_load_nolock sfs_inode index block ino_store index <= inode->blocks index == inode->blocks inode block inode dirty inode inode sfs inode sfs_bmap_load_nolock sfs_bmap_get_nolock sfs_bmap_get_nolock sfs_bmap_get_nolock sfs_bmap_load_nolock
2. sfs_bmap_truncate_nolock entry sfs_bmap_load_nolock index == inode->blocks sfs inode->blocks 0 sfs_bmap_free_nolock sfs_bmap_get_nolock sfs_bmap_get_nolock sfs_bmap_free_nolock
3. sfs_dirent_read_nolock slot entry
4. sfs_dirent_search_nolock name inode entry index entry SFS entry SFS entry->ino 0 entry block free entry SFS free entry entry

nolock inode semaphore

Inode

```
static const struct inode_ops sfs_node_fileops = {
    .vop_magic                      = VOP_MAGIC,
    .vop_open                       = sfs_openfile,
    .vop_close                      = sfs_close,
    .vop_read                       = sfs_read,
    .vop_write                      = sfs_write,
    ……
};
```

sfs_openfile sfs_close sfs_read sfs_write open close read write sfs_openfile sfs_close sfs_read sfs_write sfs_io

Inode

```
static const struct inode_ops sfs_node_dirops = {
    .vop_magic                      = VOP_MAGIC,
    .vop_open                       = sfs_opendir,
    .vop_close                      = sfs_close,
    .vop_getdirentry                = sfs_getdirentry,
    .vop_lookup                     = sfs_lookup,
    ……
};
```

sfs_opendir sys_close open close sfs_open sfs_opendir open close close sfs_getdirentry inode

10.6

Lab8

open close read write open open O_RDONLY O_WRONLY O_RDWR fd close fd

read write read C

```
count = read(filehandle, buffer, nbytes);
```

count nbytes

count -1 write

chdir opendir readdir closedir ucore opendir closedir open close readdir sys_getdirentry

open close read write sys_open sys_close sys_read sys_write readdir sys_getdirentry syscall ucore ucore file dir

user/sfs_filetest1.c main

```
int fd1 = safe_open("sfs\_filetest1", O_RDONLY);
```

ucore fd1 fd1

open->sys_open->syscall sys_open sysfile_open "sfs_filetest1" path

1. file file file_open file current->fs_struct->filemap[] fd1 file

vfs_open path inode VFS node vfs_open vfs_lookup path inode vop_open

1. “/” vfs_lookup vop_lookup SFS “/” “sfs_filetest1” vfs_lookup get_device vfs_get_bootfs “/” inode inode vfs.c inode bootfs_node init_main kern/ process/proc.c
2. vop_lookup “/” sfs_filetest1
3. file node 3 file_open “file->node=node;” current->fs_struct->filemap[fd] file node sfs_filetest1 inode fd syscall->sys_open->open->safe_open fd fd1 2 3 SFS SFS sfs_filetest1 sfs inode

SFS

vop_lookup sfs_inode.c sfs_node_dirops “.vop_lookup = sfs_lookup” sfs_lookup lab8 sfs_lookup ucore_plus

sfs_lookup node path node_store node “/” inode path sfs_filetest1 /sfs_filetest1 node_store sfs_filetest1 inode

sfs_lookup “/” path inode sfs_lookup_once sfs_filetest1 inode path sfs_filetest1 inode

sfs_lookup_once sfs_dirent_search_nolock inode SFS inode SFS inode SFS inode

```
read(fd, data, len);
```

fd len data

read->sys_read->syscall sys_read sysfile_read

1) 0

2) buffer kmalloc 4096 buffer

3)

[1]

buffer 4096 file_read buffer alen copy_to_user

[2] file_read

4 fd base len copied_store fd2file file filemap_acquire 1 vop_read iob pos iobuf_used(iob) filemap_release 1 0 file

SFS

vop_read sfs_read sfs_inode.c sfs_node_fileops .vop_read = sfs_read sfs_read

sfs_read sfs_io node inode iob write 0 1 0 inode sfs sin sfs_io_nolock iobuf_skip iobuf

sfs_io_nolock sfs_buf_op = sfs_rbuf,sfs_block_op = sfs_rblock sfs_bmap_load_nolock blkno inode sfs_rbuf sfs_rblock sfs_rblock sfs_rbuf offset + alen > din->fileinfo.size alen offset + alen dirty

sfs_bmap_load_nolock sfs_inode index block ino_store sfs_bmap_get_nolock sfs_rbuf sfs_rblock sfs_rwblock_nolock sfs_rwblock_nolock dop_io->disk0_io->disk0_read_blks_nolock->ide_read_secs

10.7

Makefile Makefile LAB8 := -DLAB8_EX1 -DLAB8_EX2(13)

```
LAB1    := -DLAB1_EX4 # -D_SHOW_100_TICKS -D_SHOW_SERIAL_INPUT
LAB2    := -DLAB2_EX1 -DLAB2_EX2 -DLAB2_EX3
LAB3    := -DLAB3_EX1 -DLAB3_EX2
LAB4    := -DLAB4_EX1 -DLAB4_EX2
LAB5    := -DLAB5_EX1 -DLAB5_EX2
LAB6    := -DLAB6_EX2
LAB7    := -DLAB7_EX1 # -D_SHOW_PHI
LAB8    := -DLAB8_EX1 -DLAB8_EX2
```

```
make

make qemu -j 16
```

```
chenyu$ make qemu -j 16
(THU.CST) os is loading ...

Special kernel symbols:
  entry  0xA00000A0 (phys)
  etext 0xA0021000 (phys)
  edata 0xA0251470 (phys)
  end   0xA0254750 (phys)
Kernel executable memory footprint: 2254KB
memory management: default_pmm_manager
memory map:
    [A0000000, A2000000]

freemem start at: A0295000
free pages: 00001D6B
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: stride_scheduler
proc_init succeeded
Initrd: 0xa005d3d0 - 0xa02513cf, size: 0x001f4000, magic: 0x2f8dbe2a
ramdisk_init(): initrd found, magic: 0x2f8dbe2a, 0x00000fa0 secs
sfs: mount: 'simple file system' (318/182/500)
vfs: mount disk0.
kernel_execve: pid = 2, name = "sh".
user sh is running!!!
$
```

10.8

lab8 lab2 2 “LAB8” challenge “LAB8” “YOUR CODE” - markdown - - ucore_lab - OS - OS

1:

sfs_inode.c sfs_io_nolock

”UNIX PIPE “

2:

proc.c load_icode make qemu -j 16 sh sh ”ls”,”hello” sfs

”UNIX “

ucore OS lab1-lab8!

11. Chiplab

11.1

OS Chiplab Chiplab QEMU I Cache D Cache

Nexys 4 DDR Chiplab bit PMON

11.2

1. `loongarch32r-linux-gnusf-strip` elf PMON

ucore-loongarch32 obj

```
loongarch32r-linux-gnusf-strip ucore-kernel-initrd
```

2. PMON Bootloader SPI Flash

> ⚠ **Warning**
>
> Nexys 4 DDR SPI Flash

1. Vivado Hardware Manager
2. PMON

   Chiplab PMON
3. Hardware Manager FPGA `Add Configuration Memory Device`
4. `s25fl128sxxxxxx0-spi-x1_x2_x4`
5. PMON `gzrom.bin`

3. Vivado Chiplab FPGA bit

Vivado

```
git clone https://gitee.com/cyyself/chiplab.git -b n4ddr_with_cpu
cd chiplab
open fpga/nexys4ddr/system_run/system_run.xpr #     open       Vivado     xpr
```

bit

> ⚠ **Warning**
>
> Vivado IP **Continue with Core Container Disabled**.

4. bit

SPI Flash PMON

5.

Chiplab 4.3 4.5 Nexys 4 DDR NAND Flash

- USB 115200 8n1
-
- PMON IP

  IP PC 169.254.0.1

  PMON IP

```
ifconfig dmfe0 169.254.0.2
ping 169.254.0.1
```

- tftp

- Windows Tftpd64
- Linux tftpd-hpa Ubuntu Wiki

  WSL2 WSL2 NAT TFTP UDP UDP Host Windows

  Lab0 tftp

- PMON

  PMON

```
load tftp://169.254.0.1/ucore-kernel-initrd
g
```

  uCore Bootloader Kernel command line Linux g

11.3

1. Chiplab N4DDR