作业四-MINIX 源代码分析

学生姓名	王雅蓉
学　　号	8206200602
专业班级	计科 2003
指导教师	沈海澜
学　　院	计算机学院
完成时间	2023 年 06 月 22 日

计算机学院

2023 年 06 月

目录

一 MINIX 系统介绍

MINIX3 是一个免费、开源的操作系统，设计目标是实现高可靠性、灵活性及安全性 1。MINIX3 主要的特点是基于微内核设计，只将必要的功能放到内核中，如进程调度、进程间通信、时钟管理和系统任务等。操作系统的其余部分作为一些独立的、受保护的进程在用户模式下运行。MINIX3 和 LINUX 二者的主要区别在于 LINUX 采用的是单体内核设计，将大量功能集成到了内核中。Minix 原来是荷兰阿姆斯特丹的 Vrije 大学计算机科学系的安德鲁 •塔能鲍姆（Andrew S. Tanenbaum ）教授所开发的一个类 UNIX 操作系统，开发初衷是方便教学使用（因为 AT&T 推出 Version 7 Unix 之后，将 Unix 源码进行了私有化）。Minix 全部的源代码共约 12,000 行，并置于他的著作 Operating Systems: Design and Implementation(ISBN 0-13-637331-3)的附录里作为范例。Minix 的系统要求在当时来说非常简单，只要三片磁片就可以启动。

二 源码解释

在学习过程中我主要学习了 MINIX3 的进程间通信需要使用的三条原语：send(dest, &message)用来向 dest 进程发送一条消息，receive(source, &message)用来接收一条来自 source 进程的消息，sendrev(src_dst, &message)用来发送一条消息，并等待一个进程的应答。在本实验中，我主要阅读了两个主要函数，分别是 mini_send 和 mini_receive。

1、mini_send: 对于发送消息的进程来说，如果 dest 进程正在等待发送进程的消息，或者等待任何消息，那么就通过内核将消息复制到接收进程；否则就将发送进程阻塞。

```
PRIVATE int mini_send(caller_ptr, dst, m_ptr, flags)
    if ( (dst_ptr->p_rts_flags & (RECEIVING | SENDING)) == RECEIVING &&
          (dst_ptr->p_getfrom == ANY || dst_ptr->p_getfrom == caller_ptr->p_nr))
{
    /* Destination is indeed waiting for this message. */
    CopyMess(caller_ptr->p_nr, caller_ptr, m_ptr, dst_ptr,
        dst_ptr->p_messbuf);
    if ((dst_ptr->p_rts_flags &= ~RECEIVING) == 0) enqueue(dst_ptr);
    } else if ( ! (flags & NON_BLOCKING)) {
    /* Destination is not waiting.  Block and dequeue caller. */
    caller_ptr->p_messbuf = m_ptr;
    if (caller_ptr->p_rts_flags == 0) dequeue(caller_ptr);
    caller_ptr->p_rts_flags |= SENDING;
    caller_ptr->p_sendto = dst;
    /* Process is now blocked.  Put in on the destination's queue. */
    xpp = &dst_ptr->p_caller_q;         /* find end of list */
    while (*xpp != NIL_PROC) xpp = &(*xpp)->p_q_link;
    *xpp = caller_ptr;                  /* add caller to end */
    caller_ptr->p_q_link = NIL_PROC;    /* mark new end of list */
```

```
    }
```

2、mini_receive:对于接收消息的进程来说，它首先会检查通知，然后才是向本进程发送的消息，如果没有等待接收的消息或者通知，就将自己阻塞。

```
251:    PRIVATE int mini_receive(caller_ptr, src, m_ptr, flags)
......
    /* Check to see if a message from desired source is already available.
    * The caller's SENDING flag may be set if SENDREC couldn't send.
    * set, the process should be blocked.
    */
    if (!(caller_ptr->p_rts_flags & SENDING)) {
......
        /* Found a suitable source, deliver the notification message. */
        BuildMess(&m, src_proc_nr, caller_ptr);    /* assemble message
        CopyMess(src_proc_nr, proc_addr(HARDWARE), &m, caller_ptr, m_ptr);
    /* Check caller queue. Use pointer pointers to keep code simple.
        xpp = &caller_ptr->p_caller_q;
        while (*xpp != NIL_PROC) {
            if (src == ANY || src == proc_nr(*xpp)) {
                /* Found acceptable message. Copy it and update status. */
            CopyMess((*xpp)->p_nr, *xpp, (*xpp)->p_messbuf, caller_ptr,
m_ptr);
                if (((*xpp)->p_rts_flags &= ~SENDING) == 0) enqueue(*xpp);
            *xpp = (*xpp)->p_q_link;       /* remove from queue */
            return(OK);                    /* report success */
            }
            xpp = &(*xpp)->p_q_link;           /* proceed to next */
        }
    }
```