4.4　实验 4-4：进程权限

1．实验目的

了解和熟悉 Linux 是如何进行进程的权限管理的。

2．实验要求

写一个用户程序，限制该程序的一些资源，比如进程的最大虚拟内存空间等。

3．实验步骤

下面是本实验的实验步骤。
进入本实验的参考代码目录进行交叉编译。

```
cd /home/lab466/runninglinuxkernel_4.0/rlk_lab/rlk_basic/chapter_8/lab4
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
```

编译 test 测试 app。然后把它拷贝到 runninglinuxkernel_4.0/kmodules 目录下面。

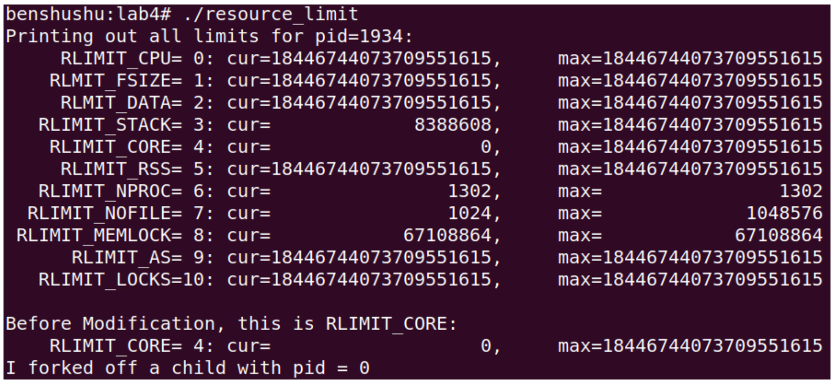
```
    arm-linux-gnueabi-gcc resource_limit.c -o resource_limit  --static
    cp resource_limit /home/lab466/runninglinuxkernel_4.0/kmodules
```

启动 QEMU+runninglinuxkernel。最好另外开一个窗口，运行：

```
    sudo su
    cd /home/lab466/runninglinuxkernel_4.0
    sh run.sh arm32
```

运行测试程序。

```
cd /mnt
./resource_limit
```


```
benshushu:lab4# ./resource_limit
Printing out all limits for pid=1934:
     RLIMIT_CPU= 0: cur=18446744073709551615,     max=18446744073709551615
    RLMIT_FSIZE= 1: cur=18446744073709551615,     max=18446744073709551615
     RLMIT_DATA= 2: cur=18446744073709551615,     max=18446744073709551615
   RLIMIT_STACK= 3: cur=             8388608,     max=18446744073709551615
    RLIMIT_CORE= 4: cur=                   0,     max=18446744073709551615
     RLIMIT_RSS= 5: cur=18446744073709551615,     max=18446744073709551615
   RLIMIT_NPROC= 6: cur=                1302,     max=                1302
  RLIMIT_NOFILE= 7: cur=                1024,     max=             1048576
 RLIMIT_MEMLOCK= 8: cur=            67108864,     max=            67108864
      RLIMIT_AS= 9: cur=18446744073709551615,     max=18446744073709551615
   RLIMIT_LOCKS=10: cur=18446744073709551615,     max=18446744073709551615

Before Modification, this is RLIMIT_CORE:
    RLIMIT_CORE= 4: cur=                   0,     max=18446744073709551615
I forked off a child with pid = 0
```

4. 实验参考代码

```c
#include <sys/time.h>
#include <sys/resource.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
#include <errno.h>

#define DEATH(mess) { perror(mess); exit(errno); }

void do_limit(int limit, const char *limit_string, struct rlimit *rlim)
{
   if (getrlimit(limit, rlim))
        fprintf(stderr, "Failed in getrlimit\n");
   printf("%15s=%2d: cur=%20lu,    max=%20lu\n", limit_string,
          limit, rlim->rlim_cur, rlim->rlim_max);
}

void print_limits(void)
{
   struct rlimit rlim;
   do_limit(RLIMIT_CPU, "RLIMIT_CPU", &rlim);
   do_limit(RLIMIT_FSIZE, "RLMIT_FSIZE", &rlim);
   do_limit(RLIMIT_DATA, "RLMIT_DATA", &rlim);
   do_limit(RLIMIT_STACK, "RLIMIT_STACK", &rlim);
   do_limit(RLIMIT_CORE, "RLIMIT_CORE", &rlim);
   do_limit(RLIMIT_RSS, "RLIMIT_RSS", &rlim);
   do_limit(RLIMIT_NPROC, "RLIMIT_NPROC", &rlim);
   do_limit(RLIMIT_NOFILE, "RLIMIT_NOFILE", &rlim);
   do_limit(RLIMIT_MEMLOCK, "RLIMIT_MEMLOCK", &rlim);
   do_limit(RLIMIT_AS, "RLIMIT_AS", &rlim);
   do_limit(RLIMIT_LOCKS, "RLIMIT_LOCKS", &rlim);
}

void print_rusage(int who)
{
   struct rusage usage;
   if (getrusage(who, &usage))
        DEATH("getrusage failed");

   if (who == RUSAGE_SELF)
        printf("For RUSAGE_SELF\n");
   if (who == RUSAGE_CHILDREN)
```

```
        printf("\nFor RUSAGE_CHILDREN\n");

    printf
       ("ru_utime.tv_sec, ru_utime.tv_usec = %4d  %4d (user time used)\n",
        (int)usage.ru_utime.tv_sec, (int)usage.ru_utime.tv_usec);
    printf
       ("ru_stime.tv_sec, ru_stime.tv_usec = %4d  %4d (system time
used)\n",
        (int)usage.ru_stime.tv_sec, (int)usage.ru_stime.tv_usec);
    printf("ru_maxrss =  %4ld (max resident set size)\n", usage.ru_maxrss);
    printf("ru_ixrss =   %4ld (integral shared memory size)\n",
          usage.ru_ixrss);
    printf("ru_idrss =   %4ld (integral unshared data size)\n",
          usage.ru_idrss);
    printf("ru_isrss =   %4ld (integral unshared stack size)\n",
          usage.ru_isrss);
    printf("ru_minflt =  %4ld (page reclaims)\n", usage.ru_minflt);
    printf("ru_majflt =  %4ld (page faults)\n", usage.ru_majflt);
    printf("ru_nswap =   %4ld (swaps)\n", usage.ru_nswap);
    printf("ru_inblock = %4ld (block input operations)\n",
          usage.ru_inblock);
    printf("ru_oublock = %4ld (block output operations)\n",
          usage.ru_oublock);
    printf("ru_msgsnd =  %4ld (messages sent)\n", usage.ru_msgsnd);
    printf("ru_msgrcv =  %4ld (messages received)\n", usage.ru_msgrcv);
    printf("ru_nsignals= %4ld (signals received)\n", usage.ru_nsignals);
    printf("ru_nvcsw=    %4ld (voluntary context switches)\n",
          usage.ru_nvcsw);
    printf("ru_nivcsw=   %4ld (involuntary context switches)\n",
          usage.ru_nivcsw);
}

int main(int argc, char *argv[])
{
    struct rlimit rlim;
    pid_t pid = 0;
    int status = 0, nchildren = 3, i;

    /* Print out all limits */

    printf("Printing out all limits for pid=%d:\n", getpid());
    print_limits();

    /* change and printout the limit for core file size */

    printf("\nBefore Modification, this is RLIMIT_CORE:\n");
    do_limit(RLIMIT_CORE, "RLIMIT_CORE", &rlim);
    rlim.rlim_cur = 8 * 1024 * 1024;
    printf("I forked off a child with pid = %d\n", (int)pid);

    setrlimit(RLIMIT_CORE, &rlim);
    printf("\nAfter  Modification, this is RLIMIT_CORE:\n");
    do_limit(RLIMIT_CORE, "RLIMIT_CORE", &rlim);

    /* fork off the nchildren */

    fflush(stdout);
    for (i = 0; i < nchildren; i++) {
        pid = fork();
        if (pid < 0)
            DEATH("Failed in fork");
        if (pid == 0) {   /* any child */
            printf("\nIn child pid= %d this is RLIMIT_CORE:\n",
                  (int)getpid());
          do_limit(RLIMIT_CORE, "RLIMIT_CORE", &rlim);
          fflush(stdout);
          sleep(3);
          exit(EXIT_SUCCESS);
      }
  }

  while (pid > 0) { /* parent */
      pid = wait(&status);
      printf("Parent got return on pid=%dn\n", (int)pid);
  }

  printf(" ************************************************** \n");
  print_rusage(RUSAGE_SELF);
  print_rusage(RUSAGE_CHILDREN);

  exit(EXIT_SUCCESS);
}
```