```
#include<linux/kernel.h>
#include<linux/init.h>
#include<linux/sched.h>
#include<linux/syscalls.h>

#include "processInfo.h"

asmlinkage long sys_listProcessInfo(void) {
    struct task_struct *proces;

    for_each_process(proces) {

    printk(
      "Process: %s\n \
       PID_Number: %ld\n \
       Process State: %ld\n \
       Priority: %ld\n \
       RT_Priority: %ld\n \
       Static Priority: %ld\n \
       Normal Priority: %ld\n", \
       proces->comm, \
       (long)task_pid_nr(proces), \
       (long)proces->state, \
       (long)proces->prio, \
       (long)proces->rt_priority, \
       (long)proces->static_prio, \
       (long)proces->normal_prio \
    );


   if(proces->parent)
      printk(
        "Parent process: %s, \
         PID_Number: %ld", \
         proces->parent->comm, \
         (long)task_pid_nr(proces-
>parent) \
      );

   printk("\n\n");

  }

  return 0;
}
```

Testing the system call:

To test the system call write a simple '**test.c**' function (it can be placed in any directory) as follows:

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>

int main()
{
    printf("Invoking
'listProcessInfo' system call");

    long int ret_status =
syscall(323); // 323 is the syscall
number

    if(ret_status == 0)
        printf("System call
'listProcessInfo' executed correctly.
Use dmesg to check processInfo\n");

    else
        printf("System call
'listProcessInfo' did not execute as
expected\n");

     return 0;
}
```