Change 1: ***stat.h***: Define the structure of all types of trap that we need to statistic count

```
struct trap_statis_s {
    int T_ALL_TRAPS;
    int T_DIVIDE;
    int T_DEBUG;
    int T_NMI;
    int T_BRKPT;
    int T_OFLOW ;
    int T_BOUND;
    int T_ILLOP;
    int T_DEVICE;
    int T_DBLFLT;
    int T_COPROC;
    int T_TSS;
    int T_SEGNP;
    int T_STACK;
    int T_GPFLT;
    int T_PGFLT;
    int T_RES;
    int T_FPERR;
    int T_ALIGN;
    int T_MCHK;
    int T_SIMDERR;
    int T_SYSCALL;
    int T_DEFAULT;
    int T_IRQ_TIMER;
    int T_IRQ_KBD;
    int T_IRQ_COM1;
    int T_IRQ_IDE;
    int T_IRQ_ERROR;
    int T_IRQ_SPURIOUS;
};
```

Change 2: ***user.h***: Define countTraps()

```
int countTraps(struct trap_statis_s *);
```

Change 3: Add SYSCALL in ***usys.S***

```
SYSCALL(countTraps)
```

Change 4: Define the system call number in ***syscall.h***

```
#define SYS_countTraps  22
```

Change 5: ***syscall.c***: There is a mapping of SYS_countTraps to sys_countTraps in the int (*syscalls[])(void) table

```
static int (*syscalls[])(void) = {
...
[SYS_countTraps]   sys_countTraps,
};
```

Change 6: In ***trap.c***, define the structure **trap_kernel_statis_s** for statistics and the function **trap_get_statis** that returns the statistics

```
struct trap_kernel_statis_s {
    int T_ALL_TRAPS;
    int T_DIVIDE;
    int T_DEBUG;
    int T_NMI;
    int T_BRKPT;
    int T_OFLOW ;
    int T_BOUND;
    int T_ILLOP;
    int T_DEVICE;
    int T_DBLFLT;
    int T_COPROC;
    int T_TSS;
    int T_SEGNP;
    int T_STACK;
    int T_GPFLT;
    int T_PGFLT;
    int T_RES;
    int T_FPERR;
    int T_ALIGN;
    int T_MCHK;
    int T_SIMDERR;
    int T_SYSCALL;
    int T_DEFAULT;
    int T_IRQ_TIMER;
    int T_IRQ_KBD;
    int T_IRQ_COM1;
    int T_IRQ_IDE;
    int T_IRQ_ERROR;
    int T_IRQ_SPURIOUS;
};
```

```
struct trap_kernel_statis_s g_kernel_trap_statis;
int trap_get_statis(void *data)
{
    memmove(data, &g_kernel_trap_statis, sizeof(g_kernel_trap_statis));

    return g_kernel_trap_statis.T_ALL_TRAPS;
}
```

Change 7: Create new file ***countTraps.c*** to test

```c
#include "param.h"
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fs.h"
#include "fcntl.h"
#include "syscall.h"
#include "traps.h"

void process_traps_conut_test(void)
{
    struct trap_statis_s cnt;
    int ret;
    int pid;

    pid = getpid();
    ret = countTraps(&cnt);

    printf(1, "get pid %d ALL Traps %d\n", pid, ret);
    if (cnt.TRAP_DIVIDE)
    printf(1, "get pid %d divide error %d\n", pid, cnt.TRAP_DIVIDE);
    if (cnt.TRAP_DEBUG)
    printf(1, "get pid %d debug exception %d\n", pid, cnt.TRAP_DEBUG);
    if (cnt.TRAP_NMI)
    printf(1, "get pid %d non-maskable interrupt %d\n", pid,
cnt.TRAP_NMI);
    if (cnt.TRAP_BRKPTRAP)
    printf(1, "get pid %d breakpoint %d\n", pid, cnt.TRAP_BRKPTRAP);
    if (cnt.TRAP_OFLOW)
    printf(1, "get pid %d overflow %d\n", pid, cnt.TRAP_OFLOW );
    if (cnt.TRAP_BOUND)
    printf(1, "get pid %d bounds check %d\n", pid, cnt.TRAP_BOUND);
    if (cnt.TRAP_ILLOP)
    printf(1, "get pid %d illegal opcode %d\n", pid, cnt.TRAP_ILLOP);
    if (cnt.TRAP_DEVICE)
    printf(1, "get pid %d device not available %d\n", pid,
cnt.TRAP_DEVICE);
    if (cnt.TRAP_DBLFLTRAP)
    printf(1, "get pid %d double fault %d\n", pid, cnt.TRAP_DBLFLTRAP);
    if (cnt.TRAP_COPROC)
    printf(1, "get pid %d reserved (not used since 486) %d\n", pid,
cnt.TRAP_COPROC);
    if (cnt.TRAP_TRAPSS)
    printf(1, "get pid %d invalid task switch segment %d\n", pid,
```

```
cnt.TRAP_TRAPSS);
    if (cnt.TRAP_SEGNP)
    printf(1, "get pid %d segment not present %d\n", pid,
cnt.TRAP_SEGNP);
    if (cnt.TRAP_STRAPACK)
    printf(1, "get pid %d stack exception %d\n", pid,
cnt.TRAP_STRAPACK);
    if (cnt.TRAP_GPFLTRAP)
    printf(1, "get pid %d general protection fault %d\n", pid,
cnt.TRAP_GPFLTRAP);
    if (cnt.TRAP_PGFLTRAP)
    printf(1, "get pid %d page fault %d\n", pid, cnt.TRAP_PGFLTRAP);
    if (cnt.TRAP_RES)
    printf(1, "get pid %d reserved %d\n", pid, cnt.TRAP_RES);
    if (cnt.TRAP_FPERR)
    printf(1, "get pid %d floating point error %d\n", pid,
cnt.TRAP_FPERR);
    if (cnt.TRAP_ALIGN)
    printf(1, "get pid %d aligment check %d\n", pid, cnt.TRAP_ALIGN);
    if (cnt.TRAP_MCHK)
    printf(1, "get pid %d machine check %d\n", pid, cnt.TRAP_MCHK);
    if (cnt.TRAP_SIMDERR)
    printf(1, "get pid %d SIMD floating point error %d\n", pid,
cnt.TRAP_SIMDERR);
    if (cnt.TRAP_SYSCALL)
    printf(1, "get pid %d system call %d\n", pid, cnt.TRAP_SYSCALL);
    if (cnt.TRAP_DEFAULTRAP)
    printf(1, "get pid %d catchall %d\n", pid, cnt.TRAP_DEFAULTRAP);
    if (cnt.TRAP_IRQ_TRAPIMER)
    printf(1, "get pid %d IRQ TIMER %d\n", pid, cnt.TRAP_IRQ_TRAPIMER);
    if (cnt.TRAP_IRQ_KBD)
    printf(1, "get pid %d IRQ KBD %d\n", pid, cnt.TRAP_IRQ_KBD);
    if (cnt.TRAP_IRQ_COM1)
    printf(1, "get pid %d IRQ COM1 %d\n", pid, cnt.TRAP_IRQ_COM1);
    if (cnt.TRAP_IRQ_IDE)
    printf(1, "get pid %d IRQ IDE %d\n", pid, cnt.TRAP_IRQ_IDE);
    if (cnt.TRAP_IRQ_ERROR)
    printf(1, "get pid %d IRQ ERROR %d\n", pid, cnt.TRAP_IRQ_ERROR);
    if (cnt.TRAP_IRQ_SPURIOUS)
    printf(1, "get pid %d IRQ SPURIOUS %d\n", pid,
cnt.TRAP_IRQ_SPURIOUS);

    return;
}
```

```
int
main(int argc, char *argv[])
{
    printf(1, "countTraps test 1\n");
    process_traps_conut_test();

    sleep(1);

    printf(1, "countTraps test 2\n");
    process_traps_conut_test();

    exit();
}
```

Change 8: Add countTraps program to UPROGS in ***Makefile***

```
UPROGS=\
    ...
    _countTraps\
```