```
int color;

#include "type.h"
#include "string.c"
// #include "queue.c"  // use provided queue.obj  during linking
// #include "kbd.c"    // use provided kbd.obj    during linking
#include "vid.c"
#include "exceptions.c"
#include "kernel.c"
#include "wait.c"
#include "pipe.c"
#include "pv.c"
#include "uart.c"



void copy_vectors(void) {
    extern u32 vectors_start;
    extern u32 vectors_end;
    u32 *vectors_src = &vectors_start;
    u32 *vectors_dst = (u32 *)0;

    while(vectors_src < &vectors_end)
       *vectors_dst++ = *vectors_src++;
}
int kprintf(char *fmt, ...);

void IRQ_handler()
{
    int vicstatus, sicstatus;
    int ustatus, kstatus;

    // read VIC status register to find out which interrupt
    vicstatus = VIC_STATUS; // VIC_STATUS=0x10140000=status reg
    sicstatus = SIC_STATUS;
    if (vicstatus & 0x80000000){
       if (sicstatus & 0x08){
          kbd_handler();
       }
    }
}

int body();

int pipe_writer()
{
  struct uart *up = &uart[0];
  char line[128];
  while(1)
  {
    kprintf("Enter a line for task1 to get: ");
    kprintf("task%d waits for line from UART0\n", running->pid);
    ugets(up, line);//gets user input
    uprints(up, "\r\n");
    printf("task%d writes line= [%s] to pipe\n", running->pid, line);
    write_pipe(kpipe, line, strlen(line));//writes the data to the pipe for the proc
to send
  }
}

int pipe_reader()
{
  char line[128];
  int i, n;
  while(1)
  {
    printf("task%d read n=%d bytes from pipe: [", running->pid, n);
    n = read_pipe(kpipe, line, 20);//read the data coming from the pipe
    printf("task%d rad n = bytes from pipe: [", running->pid, n);

    for(i = 0; i<n; i++)
    {
      kputc(line[i]);//print data
    }

    printf("\n");
  }
}

int startup()
{
    int pid, status;
  printf("P1 running: create pipe and writer reader processes\n");
  kpipe = create_pipe();
  kfork((int)pipe_writer, 1);//forks one proc to handle the writer
  kfork((int)pipe_reader, 1);//forks another proc to handle the reader
  printf("P1 waits for ZOMBIE child\n");
  while(1){
    pid = kwait(&status);
    if (pid < 0){
      printf("no more child, P1 loops\n");
      while(1);
    }
    printf("P1 buried a ZOMBIE child %d\n", pid);
  }

}

int main()
{
  fbuf_init();
  kprintf("Welcome to my version of WANIX\n");
  kbd_init();


    /* enable SIC interrupts */
   VIC_INTENABLE |= (1<<31); // SIC to VIC's IRQ31
   /* enable KBD IRQ */
   SIC_INTENABLE = (1<<3); // KBD int=bit3 on SIC
   SIC_ENSET = (1<<3);  // KBD int=3 on SIC


  kernel_init();
  kfork((int) startup, 1);

  while(1)
  {
     printf("P0 switch process\n");
     while(!readyQueue);
     tswitch();
  }

}
```