汇编语言程序设计
第二次作业

2017011620 计 73 李家昊

2019 年 8 月 26 日

Question 1 & 2

C 函数返回 struct 类型是如何实现的？

C 函数是如何传入 struct 类型参数的？

请针对讲义“C 与汇编语言-4”中最后两张 slides 的内容，回答上述问题。具体要求：语言描述的同时请画出相应的程序栈的内容示意图。

Answer 1 & 2

返回 struct 类型的实现方法：对于标记为`__cdecl` 的函数，由调用者开辟空间，被调用者在栈顶指针%rsp 的上方插入 struct 类型的数据，将%rax 指向 struct 数据的头部，然后返回给调用者。函数返回时，程序栈如下图：

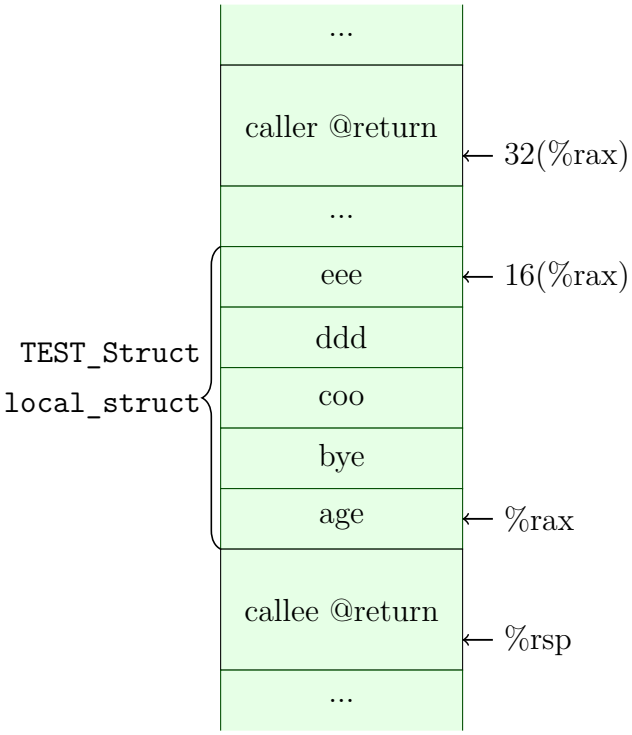


图 1: 返回 struct 类型的实现方法

传入 struct 类型参数的实现方法：由调用者开辟空间，在栈上插入 struct 类型的数据，被调用者通过%rsp 相对寻址，访问 struct 类型的数据。函数调用时，程序栈如下图：

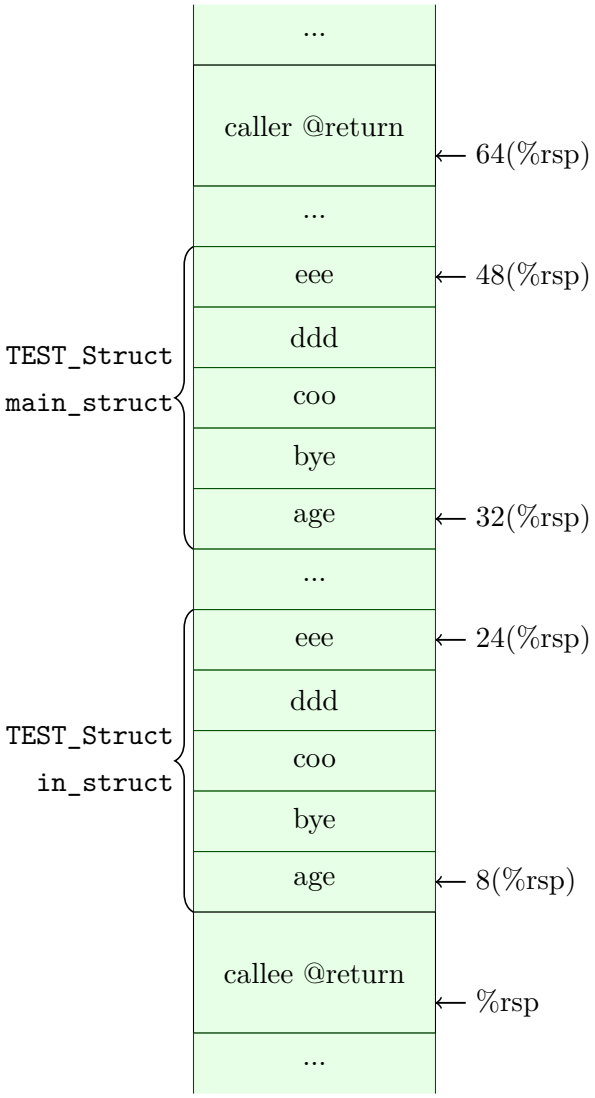


图 2: 传入 struct 类型参数的实现方法

Question 3

有如下 C 代码（X86-64 Linux 系统），编译成对应的汇编代码，请对照两边代码给出 M、N 的值。同时请在汇编代码段中标出哪些语句的作用是直接将源结构的字段复制给目标结构（每一个字段的复制指令都需分别标出）。

```
typedef struct typeTAG
{
    double  attribute_3;
    char    attribute_2;
    int     attribute_1;
} TAG;

TAG mat1[M][N];
```

```
TAG mat2[N][M];
int copy_element(int i, int j)
{
    mat1[i][j] = mat2[j][i];
    return 0;
}
```

Answer 3

经过对照得出：M 的值为 7，N 的值为 13。下面标出每个字段的复制指令：

```
copy_element:
    pushq   %rbp
    movq    %rsp, %rbp
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -8(%rbp), %eax
    movslq  %eax, %rcx
    movl    -4(%rbp), %eax
    movslq  %eax, %rdx
    movq    %rdx, %rax
    addq    %rax, %rax
    addq    %rdx, %rax
    salq    $2, %rax
    addq    %rdx, %rax
    addq    %rcx, %rax
    salq    $4, %rax
    leaq    mat1(%rax), %rcx
    movl    -4(%rbp), %eax
    movslq  %eax, %rsi
    movl    -8(%rbp), %eax
    movslq  %eax, %rdx
    movq    %rdx, %rax
    salq    $3, %rax
    subq    %rdx, %rax
    addq    %rsi, %rax
    salq    $4, %rax
    addq    $mat2, %rax

    # move mat2[j][i].attribute_2 & mat2[j][i].attribute_1 to %rdx
    movq    8(%rax), %rdx
    # move mat2[j][i].attribute_3 to %rax
    movq    (%rax), %rax
```

```
    # move %rax to mat1[i][j].attrbute_3
    movq    %rax, (%rcx)
    # move %rdx to mat1[i][j].attribute_2 & mat1[i][j].attribute_1
    movq    %rdx, 8(%rcx)

    movl    $0, %eax
    popq    %rbp
    ret
```

Question 4

设计编写汇编程序，要求如下：

（1）程序命令行输入两个文件名，都是文本文件，名字分别为 input.txt 与 output_line_num.txt。要求是统计第一个文件中的文字行数，"LF"（即换行符，ASCII 码为 0x0A）为分割行的标记，如果最后的文字直到文件末尾也没有 LF 的话，也算做一行；然后将行数以字符串形式写入第二个文件。

（2）要求在程序内实现一个名为 getline 的函数，带三个参数，第一个为读取的正文文件的文件描述符，第二个是存放一行文字（包含结尾的 0，但不包括 LF）的内存地址，第三个是该块内存的可用长度。返回值是读入的文字的长度（不包括 0 与 LF），返回-1 表示调用失败，如果已达到文件结尾，可返回 0。

要求提供原码，运行截图，以及单独成文的设计说明文字。

Answer 4

源码

详见压缩包中的 def.s，utils.s，ex1.s，ex2.s，其中 def.s 定义了系统调用相关的常量，utils.s 中实现了一些常用的字符串函数，ex1.s，ex2.s 分别是第 (1),(2) 问的解答。

设计说明

首先造轮子，实现一些常用的函数，如字符串长度函数 strlen，字符串打印函数 print_string，整数转字符串函数 itoa，单字符读取函数 getchar，将这些函数放在 utils.s 文件中。

对于第 (1) 问，首先处理命令行参数，argc 存放在%rsp 所指地址，应先确保 argc 为 3，此时输入文件名存放在 argv[1]，即16(%rsp)，输出文件名存放在 argv[2]，即24(%rsp)，调用 syscall 打开文件后，可利用 getchar 函数，此函数根据传入的文件描述符，每次从文件中读取 1 个字符，若读取失败，返回 EOF(-1)，若读取成功，则返回读取的字符。在 _start 函数中，循环调用 getchar 函

数，每次调用后保存当前字符，每次遇到换行符 `0x0A`，则令计数加一，直至遇到 EOF，则跳出循环；循环结束后需要判断上一字符是否换行符，若否，令计数加一；然后调用 `itoa`，将计数从整数转换成字符串，最后调用 `print_string` 输出到文件。

对于第 (2) 问，需要实现 `getline` 函数，可以复用第 (1) 问的 `getchar` 函数，在 `getline` 函数中，循环调用 `getchar` 函数，每次令计数增一，并将读取的字符复制到目标地址，若遇到回车或 EOF，或计数达到最大可用长度，则跳出循环，并分情况返回相应的值。这里在 `_start` 函数中循环调用 `getchar`，并利用 `print_string` 将读取的字符串打印在屏幕上，用于测试和展示，直到读取失败或遇到 EOF 时返回。

具体实现请参考代码，内有充分注释，此处不再赘述。

特别说明：在第 (2) 问中，若按照题目的返回值要求，那么在读取到空行或到达文件末尾这两种情况下，返回值均为 0，无法区分。因此，这里约定：到达文件末尾时返回 -1，保证遇到空行时正常读取。

运行截图

第 (1) 问运行结果如下图

```
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ rm output_line_num.txt
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ cat -n input.txt
     1  The first line
     2  The second line
     3
     4  Several empty lines
     5
     6
     7
     8  line A
     9  line B
    10  The last line
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ as ex1.s -g -o ex1.o && ld ex1.o -o ex1 && ./ex1 input.txt output_line_num.txt
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ cat output_line_num.txt
10
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$
```

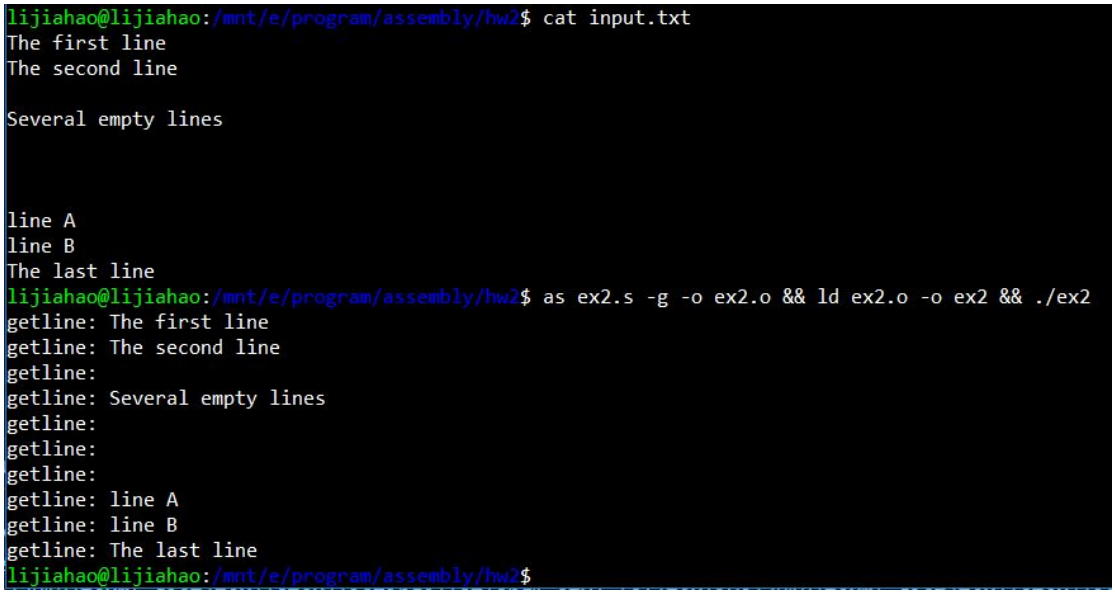
(a) Input with a trailing LF

```
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ rm output_line_num.txt
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ cat -n input.txt
     1  The first line
     2  The second line
     3
     4  Several empty lines
     5
     6
     7
     8  line A
     9  line B
    10  The last linelijiahao@lijiahao:/mnt/e/program/assembly/hw2$
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ as ex1.s -g -o ex1.o && ld ex1.o -o ex1 && ./ex1 input.txt output_line_num.txt
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$ cat output_line_num.txt
10
lijiahao@lijiahao:/mnt/e/program/assembly/hw2$
```

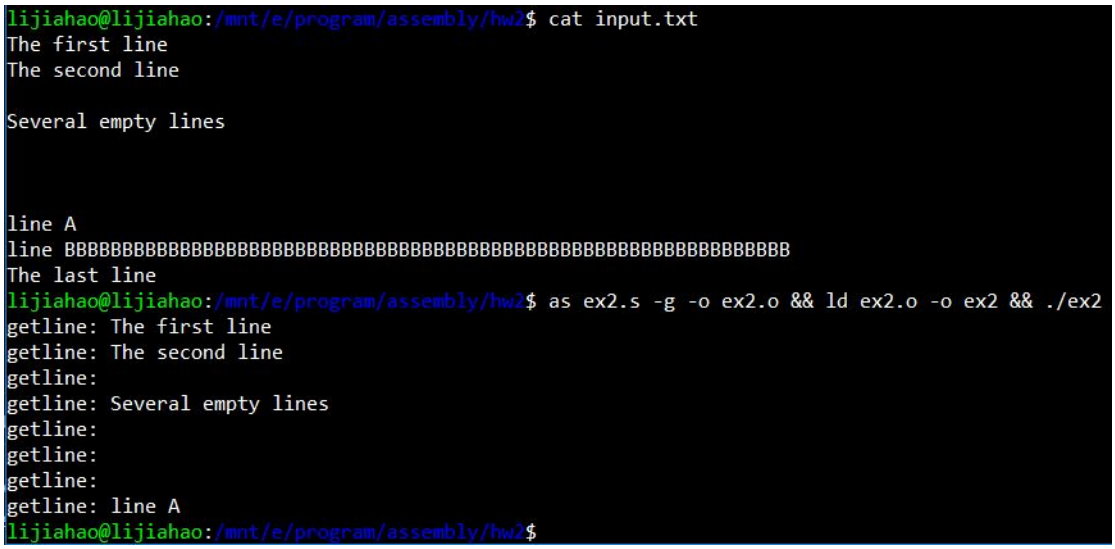
(b) Input without a trailing LF

图 3: 第 (1) 问运行截图

第 (2) 问运行结果如下图（”getline:” 前缀及末尾的换行符为单独输出，函数读取的字符串本身不含换行符）



(a) Read until EOF



(b) Break if buffer size exceeded

图 4: 第 (2) 问运行截图