

Re: Cannot return values of char variable

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2006-11/msg00558.html

- *From:* jt@xxxxxxxxxxxxx (Jens Thoms Toerring)
 - *Date:* 3 Nov 2006 12:40:31 GMT
-

Pedro Pinto <kubic62@xxxxxxxxxx> wrote:

Ok here it goes:

The problem is, when the function menu starts, i insert the information

```
CREATE TABLE [tab] col1,col2,col3
```

that starts the function `criaRespCreate` and the `buf` variable, when exported into the program, before is ok, i print it to the screen and appears well, but when returned it comes empty.....

I start the client socked, the result of the printf's:

```
sd075@lab1215-31:~/Desktop/teste/Cliente$ ./clisql 17500
Sintaxe do programa cliente:
clisql -s <endere?o_servidor> <porto_servidor>
-endere?o_servidor: (opcional) IP do servidor
-porto_servidor: porto do servidor
Insira comando:
CREATE TABLE [tab] col1,col2,col3
1 - buffer =
2 - buffer =
passei o primeiro divide com aux[0] = CREATE TABLE
passei o primeiro divide com aux[1] = tab] col1,col2,col3
passei o segundo divide com aux[0] = tab
passei o segundo divide com aux[1] = col1,col2,col3
dpx do memcpy aux[0] = tab
dpx do memcpy aux[1] = col1,col2,col3
```

Re: Cannot return values of char variable

```
buffer =
antes do strcpy - aux[1] =
argv[0] = (null)
No test to search.
```

Code -----

cliente_aux.c

```
#include "cli.h"
```

```
void syntax() {
printf("Sintaxe do programa cliente:\n");
printf(" clisql -s <endere?o_servidor>
<porto_servidor>\n");
printf(" -endere?o_servidor: (opcional) IP do
servidor\n");
printf(" -porto_servidor: porto do servidor\n");
}
```

```
void sintaxe(){
printf("Sintaxe do programa cliente:\n");
printf("CREATE TABLE [table_name] coluna1,coluna2,coluna3...\n");
printf("INSERT INTO [table_name] VALUES
(coluna1_value,coluna2_value,coluna3_value...) \n");
printf("UPDATE [table_name] SET (Coluna) WHERE (expressao)\n");
printf("SELECT [Coluna] FROM (table_name) WHERE (expressao)\n");
}
```

```
char menu(){
```

Since you seem to be trying to return a char pointer (not just a char) the function must be declared accordingly (and it doesn't take an argument, so tell the compiler):

```
char *menu( void )
```

Re: Cannot return values of char variable

```
char buf[BUFFSIZE];
char bufSaida[BUFFSIZE];
```

```
char *tmp = buf;
int cod = 0;
int id = random();
int tamanhoMsg;
```

```
printf("Insira comando:\n");
```

```
// le uma linha de input
if (fgets(buf,sizeof(buf),stdin) == NULL)
perror("fgets");
```

I guess it would make no sense to continue if fgets() failed to obtain user input, does it?

```
char comando[10];
```

Please remember that defining new variables randomly within a function is a C99 feature – you may run into trouble with this if your compiler doesn't support C99.

```
memset(comando,0,sizeof(comando));
```

This is not necessary.

```
if (sscanf(buf,"%9s",comando) < 1)
perror("scanf");
```

Again, is it a good idea to continue even though sscanf() failed? But then, what is 'comando' used for at all? It's not used anywhere below.

```
if (strncmp(buf,"CREATE",6) == 0){
cod = 11;
tamanhoMsg = criaRespCreate(cod,id, buf,bufSaida);
```

Re: Cannot return values of char variable

Re: Cannot return values of char variable

```
}
else if(strncmp(buf,"INSERT",6) == 0){
cod = 12;
//buffer =criaRespInsert(cod,id, buf);
}
else if(strncmp(buf,"UPDATE",6) == 0){
cod = 13;
//buffer =criaRespUpdate(cod,id, buf);
}
else if(strncmp(buf,"SELECT",6) == 0){
cod = 14;
//buffer =criaRespSelect(cod,id, buf);
}
else if(strncmp(buf,"QUIT",4) == 0){
exit(0);
}
else { printf("Comando Desconhecido\n");
}

return bufSaida;
```

Now, here's a very serious problem (beside the one that you defined `menu()` to return a char, not a char pointer): you return a pointer to an array that is only defined while you're within the `menu()` function. Once you have left the function you can't use the buffer anymore! Once the function has ended 'bufSaida' goes out of scope and doing anything with the return value is wrong and it can crash your program, it can seem to work flawlessly or it can seem to work at first but then result in strange results.

```
}
```

```
int criaRespCreate(int cod, int id, char *argv, char *buffer){
```

I would recommend `_not_` to use 'argv' as a variable or argument name since everybody reading your code will be confused since 'argv' (and 'argc') are the traditional names of the arguments `main()` is invoked with.

```
int tam = 0;
int nRows = 0;
char *aux[strlen(argv)];
```

Re: Cannot return values of char variable

Some compilers will complain here since the length of that array isn't known at compile time (but since you seem to be using a C99 compiler anyway it's ok).

```
char str[]=" // ";
int h = 0;
//tamanho do caracter delimitador
int tcd = strlen(str);
```

```
// inserir codigo
```

```
memcpy(buffer,&cod,LENGTH);
```

Why would you directly copy an integer to a char buffer? That hardly makes any sense. You won't have a textual representation of the number in the buffer but some bytes that don't make any sense when the content is interpreted as a string. Moreover, you don't have a '\0' character at the end, so trying to use 'buffer' as it were a string (as you do below) is a recipe for disaster. Perhaps what you really want here is `sprintf()`?

And if you really want to copy the bytes of an integer than you should use `'sizeof(int)'` or `'sizeof cod'` instead of `LENGTH` (which is defined as 4, which can be the right size, but that's not guaranteed).

```
tam += LENGTH;
```

```
printf("1 - buffer = %s\n", buffer);
```

Here things get badly wrong: 'buffer' doesn't end in a '\0', so it's not a string, so it can't be used as an argument to `printf()` with `"%s"`. And then, having an int copied into a string doesn't make it a string representation of that value. I guess you're assuming that there's some automatic conversion happening but that's not the case.

```
// inserir caracter delimitador //
memcpy(buffer+tam,&str,strlen(str));
```

Re: Cannot return values of char variable

```
tam += tcd;
```

'buffer' still has no '\0' at the end since you didn't copy that from 'str'. Perhaps the function you are looking for is strcpy()?

```
printf("2 - buffer = %s\n", buffer);
```

And thus this is the next point where things go wrong.

```
// inserir id da mensagem  
memcpy(buffer+tam, &id, LENGTH);  
tam += LENGTH;
```

Same problem as above, copying an integer to a char buffer doesn't make any sense if you expect the char buffer to contain a representation of the int that would make sense as a string.

```
// inserir caracter delimitador //  
memcpy(buffer+tam,&str,strlen(str));  
tam += tcd;
```

```
// dividir string e inserir nome tabela  
divide(argv,aux,"");
```

Unfortunately, there's neither a declaration nor a definition of divide() in what you posted so it's impossible to say if the way you call it is correct (and if it does what you seem to expect) or what effect of 'aux' (which you use in the following) it has...

```
printf("passei o primeiro divide com aux[0] = %s\n\n", aux[0]);  
printf("passei o primeiro divide com aux[1] = %s\n\n", aux[1]);
```

```
memset(argv,0,strlen(aux));  
strcpy(argv,aux[1]);  
memset(aux,0,strlen(argv));  
divide(argv,aux,"");
```

Re: Cannot return values of char variable

```
printf("passei o segundo divide com aux[0] = %s\n\n", aux[0]);
printf("passei o segundo divide com aux[1] = %s\n\n", aux[1]);
memcpy(buffer+tam,&aux[0],strlen(aux[0]));
```

Since 'buffer' as a fixed, final size and you don't have any idea how long the string aux[0] points to is you could easily write past the end of 'buffer' if the string is too long. Let's just hope that your divide function at least works in a way that the strings the elements of 'aux' point to have an '\0' at the end, or there would be lots errors in the things to come...

```
tam += strlen(aux[0]);
```

```
printf("dpx do memcpy aux[0] = %s\n\n", aux[0]);
printf("dpx do memcpy aux[1] = %s\n\n", aux[1]);
// inserir caracter delimitador
memcpy(buffer+tam,&str,strlen(str));
tam += tcd;
printf("buffer = %s\n", buffer);
```

Since you still have no '\0' at the end of 'buffer' you still can't use it as the argument to printf() with "%s".

```
// inserir numero e nome colunas
```

```
memset(argv,0,strlen(aux[1]));
```

How do you know for sure that the amount of memory 'argv' points to is large enough to hold as many 0s as aux[1] has characters?

```
printf("antes do strcpy - aux[1] = %s\n\n", aux[1]);
```

```
strcpy(argv,aux[1]);
printf("argv[0] = %s\n", argv[0]);
```

Since what you defined as 'argv' is a char pointer, argv[0] is a char, so you can't use it with "%s", you would need "%c".

Re: Cannot return values of char variable

```
memset(aux,0,strlen(argv));
```

'aux' is an array of pointers (as many as 'argv' has characters.
If you want to zero them all, then you would need

```
memset( aux, 0, strlen(argv) * sizeof *aux );
```

But then there's also the problem that 0 is not necessarily a NULL pointer...

```
divide(argv,aux,"");
```

```
printf("passei o terceiro divide com aux[0] = %s\n\n", aux[0]);  
printf("passei o terceiro divide com aux[1] = %s\n\n", aux[1]);  
printf("passei o terceiro divide com aux[2] = %s\n\n", aux[2]);
```

```
printf("str(aux) = %d\n", strlen(aux));
```

Since 'aux' is not a string but an array of pointers calling strlen() on it is simply wrong.

```
// inserir numero de colunas  
for(h=0; aux[h]!=NULL;h++){  
    nRows++;  
}
```

Let's hope either a NULL pointer is represented by all bits 0 or your divide() function did the right thing...

```
memcpy(buffer+tam, nRows, LENGTH);  
tam += LENGTH;
```

Again, copying the bits of an int into a char array may not be what you want...

```
// inserir caracter delimitador  
memcpy(buffer+tam,&str,strlen(str));  
tam += tcd;
```

Re: Cannot return values of char variable

```
// inserir colunas

for(h=0; aux[h]!=NULL;h++){
memcpy(buffer+tam,&aux[h],strlen(aux[h]));
tam += tcd;
// inserir caracter delimitador
memcpy(buffer,&str,strlen(str));
tam += tcd;

}
return buffer;
}
```

clisql.c-----

```
#include "cli.h"
```

```
/* Funcao Main */
```

```
int main (int argc, char *argv[]) {
```

```
    syntax();
```

```
    /* Funcao que apresenta sintaxe do programa */
```

Calling a function before you have defined all variables is only working with a C99 compiler.

Re: Cannot return values of char variable

```
int sock;
int broadcast = 1;
struct sockaddr_in server;
struct sockaddr_in cli;
int porto_servidor;
char buffer[BUFSIZE];
char *tmp = buffer;
tmp = buffer;
char input[1024];

/* Limpa a estrutura */

bzero((char *)&server, sizeof(server));
bzero((char *)&cli, sizeof(cli));
```

Why not use memset()?

```
/* Limpar o buffer */

memset(buffer, 0, BUFSIZE);

input[0] = menu(buffer);
```

Do you remember? menu() was defined to return a char, but in reality it did return a pointer to a char buffer that you can't use here anymore. So whatever is stored in 'input[0]' is rather likely to be complete garbage. Luckily, you never use 'input' in the following. But then what's the reason for this assignment?

Even worse, menu() doesn't accept any arguments, but you call it with one. That should make your compiler complain loudly. And 'buffer' is never going to be set up to anything you seem to expect, so using it in the following is a bad mistake.

I am not going to comment on the use of non-standard functions like socket() etc. in the following, these are things better left for groups like comp.unix.programmer, and you have enough problems with C anyway that you should deal with first.

Re: Cannot return values of char variable

```
/* Criacao da socket */
if ((sock = socket(AF_INET,SOCK_DGRAM,0)) <0) {
perror("socket");
exit(1);
}
else {
printf("Cliente: socket criada \n");
}

/* Verificar ip */
if(strcmp(argv[1], "-s") == 0) {
printf("entrei n o verifica ip, estamos a funcionar com endereco ip
inserido\n");
inet_aton(argv[2], &server.sin_addr);
porto_servidor = atoi(argv[3]);
}
```

Checking that argc is at least 4 and that the elements of argv are really strings of the form you expect would be the RIGHT thing to do before using them.

```
else {//enviar em broadcast
printf("N?o foi fornecido o endere?o IP, a enviar em broadcast\n");
porto_servidor = atoi(argv[1]);
server.sin_addr.s_addr = htonl(INADDR_BROADCAST);
/* Activar broadcast na socket */

if (setsockopt(sock, SOL_SOCKET, SO_BROADCAST, (char*)&broadcast,
sizeof (int)) <0) {
perror("setsockopt");
exit(1);
}
}

/* verifica se a porta esta entre os valores pretendidos,
* visto o n?mero de grupo ser o 75, port vai variar entre
* 17500 e 17599 */
```

You already have been told that defining a function within another function isn't allowed in C.

Re: Cannot return values of char variable

```
void checkPort(porto_servidor) {
```

You need to supply the type of the argument (probably 'int').

```
if(porto_servidor < 17500 || porto_servidor > 17599) {
    printf("Cliente: O porto tem de ser entre 17500 e 17599!\n");
    exit(-1);
}
}
```

```
/* Definir a familia de protocolo e porto */
server.sin_family = AF_INET;
server.sin_port = htons(porto_servidor);
```

```
/* Inicio da comunicacao com o servidor */
/* Enviar a mensagem para o servidor*/
if(sendto(sock, tmp, strlen(buffer), 0, (struct sockaddr *)&server,
sizeof(server)) < 0) {
```

Let's assume 'buffer' was set up in menu() (which didn't happen, see above) via the call to criaRespCreate() (but only under certain circumstances, in most cases it would just contain garbage), and the way it's potentially would be set up there doesn't make sure there's a '\0' at its end, so using it as the argument to strlen() is not going to work.

```
perror("sendto\n");
exit(1);
}
```

```
else {
    printf("Cliente: Mensagem enviada com sucesso\n");
}
```

```
while(1){
    printf("entrei no while\n");
    /* timeout de 10 segundos */
    struct timeval timeout;
    timeout.tv_sec = 10; //segundos
    timeout.tv_usec = 0; //microsegundos
```

Re: Cannot return values of char variable

```
fd_set readfds;
int sel;
// char buf1[64];

/* limpar a fd_set */
FD_ZERO(&readfds);

/* colocar o file descriptor no fd_set */
FD_SET(sock, &readfds);
printf("vou ver agora o select\n");
if((sel = select(sock+1, &readfds, NULL, NULL, &timeout)) < 0)
perror("select\n");
else if (sel == 0){

printf("Ocorreu um timeout! Nao foi recebida nenhuma mensagem em
10s.\n");
exit(0);
}

memset(buffer,0,BUFFSIZE);
printf("vou agora ler a resposta do servidor\n");
/* ler a resposta do servidor */

if(recvfrom(sock, buffer, BUFFSIZE, 0, NULL, NULL) < 0) {
printf("entrei no recv por ser < 0\n");
perror("recvfrom\n");
exit(1);
}
else {
printf("Cliente: Mensagem recebida com sucesso\n");
printf("buffer = %s\n", buffer);
```

What makes you sure that what you received from the server has a '\0' character at the end, so you can use 'buffer' with "%s" in printf()?

```
exit(0);
}
}
```

Re: Cannot return values of char variable

You will never get here because of the infinite while loop above without any 'break' that would you get you out of it (actually the loop will only repeat as long as select() returns a negative value, i.e. indicates an error happened, in all other cases you just exit from the program).

```
/* fecho da socket */  
close(sock);  
}
```

-----cli.h-----

```
#ifndef CLI_H  
#define CLI_H
```

```
#include "projecto.h"
```

```
/* Definicao das funcoes definidas em cliente_aux.c */
```

```
void syntax();  
char menu();  
int criaRespCreate(int cod, int id, char *argv, char *buffer);
```

```
#endif
```

-----projecto.h-----

```
#ifndef PROJECTO_H  
#define PROJECTO_H
```

Re: Cannot return values of char variable

```
#define BUFFSIZE 6804
#define LENGTH 4
```

```
/* Bibliotecas necessarias */
```

```
#include <sys/socket.h>
#include <sys/types.h>
#include <stdio.h>
#include <errno.h>
#include <sys/time.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
```

```
#endif
```

I can only recommend that you start with something more simple, like some functions for manipulating strings and non-string memory, testing them carefully, in order to get a better idea what's the difference.

Try to figure out when you have to allocate memory and when not, how to return memory from functions etc. before you embark on a complex project involve both dealing with difficult user input, communication with a server (and getting the protocol right) etc.

Regards, Jens

--

\ Jens Thoms Toerring ___ jt@xxxxxxxxxxx

_____ <http://toerring.de>

.