

## Re: unusual behaviour of program under linux....

**Source:** <http://unix.derkeiler.com/Newsgroups/comp.unix.programmer/2003-06/0710.html>

---

**From:** seemanta dutta (*seemanta\_18\_at\_yahoo.com*)

**Date:** 06/19/03

Date: 19 Jun 2003 14:35:27 -0700

greetings linux gurus...

what u have demonstrated by ur overwhelming response has left me utterly speechless and grateful to the vast unix community and their willingness to help bungling beginners...  
thanks for ur patience in pointing out mistakes of beginners...

i have a few more questions to ask:

1. i feel ashamed, but what exactly is an uninitialised pointer?  
is

```
int *localVar;  
*localVar = 3 ;  
an uninitialised pointer?
```

or is this is an \*initialised\* pointer?

```
int *localVar;  
localVar = malloc(sizeof int);
```

please clarify with examples of both initialised and uninitialised pointers.

2. in some of my programs i have to read from a file and write to a different file using fgets and fprintf...

what i want to ask is that when i declare my buffer for fgets as  

```
char *buff;
```

my program compiles alright but while running it gives a seg fault..i can understand this is because of a buffer overflow, but if i use a declaration like this for my buffer

```
char buff[80];
```

my seg fault disappears...

but here i am implicitly assuming my data will never be larger than 80 bytes...if my data exceeds 80 bytes there occurs another seg fault and my program exits...

is it a good practice to use limited size buffers for file i/o?

what i want to know is that is there any way to write code for file i/o ,or any i/o for that matter which shall be independent of the amount of data entered by the user or of that amount of characters present in a line of the file read by a call to fgets.

actually i am writing an assembler for 8051 microcontroller and for it i have to first finish my ground work by writing all my text procesing functions for macro substitution and line splicing etc...

it is there that i am facing the problem with limited size buffers for performing file i/o with 80 byte-sized buffers...

this question came to my mind because gcc and gas function without without any apparent segmentation fault even while processing files with very long lines.

please help...

3. in some of my programs like this..

```
int main()
{
    char deststr[20],srcstr1[20]="foo",srcstr2[20]="bar";
    deststr=strcat(srcstr1,srcstr2); // insertion of 4 given
lines made here in place of this line...
    puts(deststr);
    return 0;
}
```

but the above program compiles with the following error: t.c: In function `main':

t.c:8: incompatible types in assignment

if the man page says that strcat returns a 'char \*' and since the name of a char array implies a pointer to a char, why is this giving an error?

if i replace the declaration of deststr with char \*deststr; i no longer get any error, but on running i get a seg fault...obviously due to a buffer overflow and if i use desstr[20] i get the above error.

what to do? it is a catch22 kind of situation for me... i cannot use \*deststr because of buffer overflows,neither can i use deststr[20] for it does allow me to use strcat as i want to... and gives an error

if i insert these lines in place of the second line above, i can use deststr[20] instead...but at the expense of a few more lines of code...

and a limited buffer size of deststr[20]...

```
char temp[30];

strcpy(temp,srcstr1);
strcat(temp,srcstr2);
strcpy(deststr,temp);
```

but is it ok to involve another variable like this and use an awkward strcpy and a limited buffer size?

please advise where i am making a mistake...

4. please consider the following code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

char * foo(void);

int main()
{
    char *ptr ;
    ptr = foo();
    puts(ptr);
    return 0;
}

char * foo()
{
    char *retval; //using retval[10] causes warning...but using
*retval doesnot..
    strcpy(retval,"hello"); //why is strcpy allowed here?
should'nt a seg fault occur here?
    return retval; //using strcat also does not cause a
seg fault...

// question: is there a memory leak in this code if i add the line
'retal=malloc(7);' after retval declaration?

}
```

in the first line in foo(),if i replace \*retval with retval[20] or something like it, i get a warning saying warning: function returns address of local variable

is that an unacceptable practice to return address of an auto variable?

and in one program i was using such a variable by malloc'ig some bytes...

```
char *retval;  
retval=malloc(5);  
strcpy(retval,"hello");  
...  
return retval;
```

using malloc call in place of retval[20] also removes the above warning.

but this has put me into worries since i have read in books that malloc() calls that do not have any corresponding free() calls cause memory leaks...

so in this case i ought to call free()

but if i call free before i call return my pointer that would return a garbage pointer, is n't it?

but on the other hand i must free() my memory also...as well return my pointer to the calling function...

how to write code that would prevent memory leaks as well as won't give any warnings as shown above...

5. in a program while debugging i got a 'program returned 62 error code' from gdb.

how do i convert this error code into a meaningful string, like with errno using perror? i can collect the exit code of my program thru the wait syscall which matched with that returned by gdb, but how do i get some meaningful info from it in human readable form?

please help...

that was all...hope i did not bore u with these bombardment of silly questions... ;)

thanks in advance...

urs sincerely  
seemanta ;)