getlogin() c function returns NULL and error “No such file or directory”

I have a question regarding the getlogin() function (). I tried to get the login name of my account from the c program using this function. But the function returns a NULL. Using perror shows that the error is "No such file or directory".

I don't get what is the problem. Is there a way to get user login name in a program.

Here is a sample code:

```
#include <stdio.h>
#include <unistd.h>

int main()
{
  char *name;
  name = getlogin();
  perror("getlogin() error");
  //printf("This is the login info: %s\n", name);
  return 0;
}
```

And this is the output: `getlogin() error: No such file or directory`

Please let me know how to get this right.

Thanks.

c programming-languages

edited Jan 25 '11 at 0:43

Tim Post ♦
25.8k 13 86 152

asked Jan 24 '11 at 17:38

yaami
318 4 13

1 welcome to SO. You'll notice on this site we don't mark threads solved with [solved]. Next to everyone's answer, there's a tickbox you can use to mark the post that "answered" your question as the answer. If none did, you have two choices - post a full solution in answer to your own question or choose the answer that's closest, depending on how far apart the two are. – user257111 Jan 24 '11 at 18:57

5 Answers

`getlogin` is an unsafe and deprecated way of determining the logged-in user. It's probably trying to open a record of logged-in users, perhaps `utmp` or something. The correct way to determine the user you're running as (which might not be the same as the logged-in user, but is almost always better to use anyway) is `getpwuid(getuid())`.

answered Jan 24 '11 at 17:

R..
136k 17 213 469

3 Thank you That worked for me. Here is the sample code I used. #include <stdio.h> #include <pwd.h> int main() { char *name; struct passwd *pass; pass =

getpwuid(getuid()); name = pass->pw_name; printf("This is the login name: %s\n", name); return 0; } – yaami Jan 24 '11 at 18:09

Maybe you could edit the answer to include a properly formatted example, like yaami added? – nmz787 Feb 7 at 20:04

Here is a good link I found explaining that it may not work: getlogin

Here is a quote from it:

> Unfortunately, it is often rather easy to fool getlogin(). Sometimes it does not work at all, because some program messed up the utmp file

answered Jan 24 '11 at 17:

Nick Banks
2,611 4 26 59

It works fine for me if I comment `perror` call.

From `man` :

> getlogin() returns a pointer to a string containing the name of the user logged in on the controlling terminal of the process, or a null pointer if this information cannot be determined.'

So you should do:

```
#include <stdio.h>
#include <unistd.h>

int main()
{
  char *name;
  name = getlogin();
  if (!name)
    perror("getlogin() error");
  else
    printf("This is the login info: %s\n", name);
  return 0;
}
```

answered Jan 24 '11 at 17:

Elalfer
4,747 11 19

That is just not safe ... – Tim Post ♦ Jan 24 '11 at 17:55

What do you mean `not safe` ? Execution or system security? – Elalfer Jan 24 '11 at 18:00

System security .. (sorry for being ambiguous), It is just too easy to fiddle with / break `utmp` . It is much safer to use a function that gets this information from /etc/passwd, and less prone to breakage if `utmp` is borked. – Tim Post ♦ Jan 24 '11 at 18:03

1 I agree, but it wasn't a question about `how to trick getlogin()` – Elalfer Jan 24 '11 at 18:05

Thanks for your answer. But for me the function does not give the login name. It just gives back null always. So I used the getpwuid() function suggested by R.. and worked as I wanted. – yaami Jan 24 '11 at 18:08

According to the man page the error (ENOENT) means:

> There was no corresponding entry in the utmp-file.

answered Jan 24 '11 at 17:46

DarkDust

69.6k 10 140
180

I typically use getpwent() along with a call to geteuid() and getegid(). This gives me all of the information that I might possibly need to know (at least as far as /etc/passwd has to offer) and tells me if I'm running as setuid / setgid, which is helpful when programming defensively.

I have written several programs for my company that outright refuse to work if someone tries to setuid them and change ownership to root, or refuse to run as root if being called by a system user (www-data, nobody, etc).

As others have said, reading from `utmp` is a *very* bad idea for this purpose.

edited Jan 24 '11 at 18:02 answered Jan 24 '11 at 17:

Tim Post ♦
25.8k 13 86
152