Solutions to Assignment1

Commands.txt

1 xemacs &
2 fork xemacs, the emacs editor so the app and terminal runs at the same time
3
4 cd
5 change directory
6
7 cat ~/.bashrc > tmpfile.txt
8 redirects the output from ~/.bashrc into a tmpfile.txt
9
10 ln -s tmpfile.txt ~/tmp-alias
11 symbolic link between ln -s targe_file linkfile_name
12
13 ls -al
14 list all file and their information within the directory
15
16 chmod a+rwx tmpfile.txt
17 change mode to all users can access and set read write execute, same as chmod 777
18
19 grep bash /etc/passwd
20 search for bash within directory /etc/passwd
21
22 ps -ef | more
23 to view processes run on the system -ef option for viewing all of them
24
25 man 2 chown
26 manual for change ownership for a given file
27
28 gcc test.c 2> error-msg
29 gnu compiler output to error -msg

gcc_and_gdb.txt

1 2a)What does the -E option mean? What does the pre-processor do
2 -E means only run the preprocessor. the preprocessor used to be called macro processor that does textual transformation on its input before the processing. Any input file i s read into memory into many lines. continued line are concatenated into one line. and comments are taken out.
3
4 2b)hello.o is object file, which is compiled but not linked. Each file is compiled but not linked together. If the file is not compiled then it is just ignored.
5
6 2c)I get the a.out preprocessed, compiled, and linked
7
8 2d)Undefined symbols for architecture x86_64:
9 The error is saying that they cant find main function of the program.
10
11 2e)they are no error messages but when link the object file, i get
12 the same error as before. I am suspecting the linker final step checks
13 for the existence of the main function.
14
15 3.Debugging with gdb
16 3a)[1] 75541 segmentation fault ./hello
17 it means there is a failure condition raised by hardware with memory protection.
18
19 3b)Program received signal SIGSEGV, Segmentation fault.
20 __GI_____strtol_l_internal (nptr=0x0, endptr=endptr@entry=0x0, base=base@entry=10,
21 group=group@entry=0, loc=0x7ffff7dd4060 <_nl_global_locale>) at ../stdlib/strtol_l .c:298
22 298 ../stdlib/strtol_l.c: No such file or directory.
23 It must be strtol_l.c in stdlib that is causing the error.
24
25 3c)__GI_____strtol_l_internal (nptr=0x0, endptr=endptr@entry=0x0, base=base@entry=10,
26 group=group@entry=0, loc=0x7ffff7dd4060 <_nl_global_locale>) at ../stdlib/strtol_l .c:298
27 #1 0x00007ffff7a523f2 in __GI_strtol (nptr=<optimized out>, endptr=endptr@entry=0x0,
28 base=base@entry=10) at ../stdlib/strtol.c:108
29 #2 0x00007ffff7a4eeb0 in atoi (nptr=<optimized out>) at atoi.c:27
30 #3 0x00000000004005cb in main (argc=1, argv=0x7fffffffe138) at hello.c:13
31 The main in hello.c in line 13 and atoi in line 27 causes the error because it is expe cting an input but instead got none.
32
33 3d)(gdb) p argc
34 $1 = 1 this one argument is the file itself
35 (gdb) p argv
36 $2 = (char **) 0x7fffffffe138
37 argv is a one-dimensional array of strings. the value argv is the address to

the char of strings. In this case argv[1] is the file to be executed
38
39 e)(gdb) info args give us the info on arg c and arg v
40 (gdb) info local lists all local variables of the current stack frame
41
42 f)(gdb) p argv[0] return the directory of the executable
43 (gdb) p argv[0][1] returns 104 'h', the first character of the file directory
44 (gdb) p argv[1] return memory address 0x0
45
46 4
47 a)(gdb) p x where x is an int pointer storing a memory address
48 $2 = (int *) 0x0
49 (gdb) p y where y is value so value is printed
50 $3 = 0
51 (gdb) p *x dereference value stored at pointer 0x0, which is null
52 Cannot access memory at address 0x0
53 (gdb) p *y
54 Cannot access memory at address 0x0 because the *y is not defined
55 (gdb) p &x location where pointer x is stored
56 $4 = (int **) 0x7fffffffe038
57 (gdb) p &y location where pointer y is stored
58 $5 = (int *) 0x7fffffffe034
59
60 b) x = 0x7fffffffe044 x is now changed to address of y which is same as &y
61 *x = 1 dereferences x which is the value store at &y which is now being set to 1
62 y = 1 y is reset because x store the address of y and *x resets the y value to 1
63
64 c)(gdb) p x
65 $11 = (int *) 0x7fff0000000a because x is set 10
66 (gdb) p y
67 $12 = 1
68 (gdb) p *x
69 Cannot access memory at address 0x7fff0000000a because no value is stored here
70 (gdb) p *y
71 Cannot access memory at address 0x1 because is y is already dereferenced
72
73 d)(gdb) p /x x
74 $18 = 0x7fffffffe044 this is the address of y
75 (gdb) p /x y
76 $19 = 0xffffe044 y stores the same value as x
77 (gdb) p *x
78 $20 = 0xffffe044 deferences x which stores the address of y
79 (gdb) p *y
80 Cannot access memory at address 0xffffffffffffe044 this is still undefined.
81
82 PointersInGdb.c:4:13: warning: initialization makes integer from pointer without a cas t [enabled by default]
83 int y = x;
84

85 e) -Wall turns on all the warning flags that some users considers questionable
86
87
88 9.
89 int main(void)
90 {
91 int *x = 0;
92 int y = &x;
93 return 0;
94 }