

Ejercicio 1

El programa **strings** de Linux, busca en los archivos strings, y hace una lista de éstos.

1. Corra el programa **fortune**

```
nahuel-prado@nahuel-prado-HP-245-G8-Notebook-PC:~/Escritorio/Estudio/Arqui/Tp1/binarios$ ./fortune_64
Bienvenido a al adivinador de la fortuna! Este mensaje es muy largo, y puede ser muy molesto al usuario. Tal vez deberiamos acortarlo?
Cual es tu nonbre?: Nahuel
```

```
Tu fortuna es:
Break into jail and claim police brutality.
```

2. Ejecute **strings** utilizando como argumento el programa **fortune**. Identifique todas las fortunas que dicho programa estaría indicando.

Aparecen muchas cosas, entre ellas:

```
Break into jail and claim police brutality.
Never be led astray onto the path of virtue.
You will forget that you ever knew me.
Your society will be sought by people of taste and refinement.
You will be honored for contributing your time and skill to a worthy cause.
Expect the worst, it's the least you can do.
You may not get this fortune
```

3. ¿Cómo hace dicho programa para encontrar los strings? Implemente su propia versión.

Va a la seccion datos y la interpreta bajo la tabla ascii ?

```
#include <stdio.h>

void append(char * dest,int size, char org){
    dest[size]=org;
    dest[size+1]=0;
}

int main(int argc, char const *argv[]){
    FILE* file=fopen(argv[1],"rt");
    char toPrint[500]={0};
    if (file!=NULL){
        int amount=0;
        while (!feof(file)){
            int c=getc(file);
            if (c >= 32 && c <= 126){
                append(toPrint,amount,c);
                amount++;
            }else if (amount){
                if (amount >= 4){
                    printf("%s\n",toPrint);
                }
                toPrint[0]=0;
                amount=0;
            }
        }
        return 0;
    }
}
```

Al final no, mucho más facil, leo todo el archivo, me quedo con los caracteres imprimibles y luego con los char* de longitud mayor a 4 para asegurarme que es texto y no estoy imprimiendo letras sueltas

Ejercicio 2

Abra con el **Bless Hex Editor** y el programa **fortune**. Busque los Strings que aparecen en el programa y cámbielos.

1. Corrija los horrores de ortografía.

```
./2_1_fortune-ortografia
Bienvenido a al adivinador de la fortuna! Este mensaje es muy largo, y puede ser
muy molesto al usuario. Tal vez deberiamos acortarlo?
Cual es tu nombre?: s

Tu fortuna es:
Never be led astray onto the path of virtue.
```

File Edit View Search Tools Help

2_1_fortune-recortado x 2_1_fortune-ortografia x

```

000005d0 42 72 65 61 6B 20 69 6E 74 6F 20 6A 61 69 6C 20 61 6E 64 20 63 6C 61 69 6D 20 70 6F 6C 69 63 Break into jail and claim polic
000005ef 65 20 62 72 75 74 61 6C 69 74 79 2E 00 4E 65 76 65 72 20 62 65 20 6C 65 64 20 61 73 74 72 61 e brutality..Never be led astra
0000060e 79 20 6F 6E 74 6F 20 74 68 65 20 70 61 74 68 20 6F 66 20 76 69 72 74 75 65 2E 00 00 00 00 59 y onto the path of virtue....Y
0000062d 6F 75 20 77 69 6C 6C 20 66 6F 72 67 65 74 20 74 68 61 74 20 79 6F 75 20 65 76 65 72 20 6B 6E ou will forget that you ever kn
0000064c 65 77 20 6D 65 2E 00 00 59 6F 75 72 20 73 6F 63 69 65 74 79 20 77 69 6C 6C 20 62 65 20 73 6F ew me...Your society will be so
0000066b 75 67 68 74 20 62 79 20 70 65 6F 70 6C 65 20 6F 66 20 74 61 73 74 65 20 61 6E 64 20 72 65 66 ught by people of taste and ref
0000068a 69 6E 65 6D 65 6E 74 2E 00 00 59 6F 75 20 77 69 6C 6C 20 62 65 20 68 6F 6E 6F 72 65 64 20 66 inement...You will be honored f
000006a9 6F 72 20 63 6F 6E 74 72 69 62 75 74 69 6E 67 20 79 6F 75 72 20 74 69 6D 65 20 61 6E 64 20 73 or contributing your time and s
000006c8 6B 69 6C 6C 20 74 6F 20 61 20 77 6F 72 74 68 79 20 63 61 75 73 65 2E 00 45 78 70 65 63 74 20 kill to a worthy cause..Expect
000006e7 74 68 65 20 77 6F 72 73 74 2C 20 69 74 27 73 20 74 68 65 20 6C 65 61 73 74 20 79 6F 75 20 63 the worst, it's the least you c
00000706 61 6E 20 64 6F 2E 00 59 6F 75 20 6D 61 79 20 6E 6F 74 20 67 65 74 20 74 68 69 73 20 66 6F 72 an do..You may not get this for
00000725 74 75 6E 65 00 00 00 42 69 65 6E 76 65 6E 69 64 6F 20 61 20 61 6C 20 61 64 69 76 69 6E 61 64 tune...Bienvenido a al adivinad
00000744 6F 72 20 64 65 20 6C 61 20 66 6F 72 74 75 6E 61 21 20 45 73 74 65 20 6D 65 6E 73 61 6A 65 20 or de la fortuna! Este mensaje
00000763 65 73 20 6D 75 79 20 6C 61 72 67 6F 2C 20 79 20 70 75 65 64 65 20 73 65 72 20 6D 75 79 20 6D es muy largo, y puede ser muy m
00000782 6F 6C 65 73 74 6F 20 61 6C 20 75 73 75 61 72 69 6F 2E 20 54 61 6C 20 76 65 7A 20 64 65 62 65 olesto al usuario. Tal vez debe
000007a1 72 69 61 6D 6F 73 20 61 63 6F 72 74 61 72 6C 6F 3F 00 43 75 61 6C 20 65 73 20 74 75 20 6E 6F riamos acortarlo?.Cual es tu no
000007c0 6D 62 72 65 3F 3A 20 00 25 73 00 0A 54 75 20 66 6F 72 74 75 6E 61 20 65 73 3A 0A 20 25 73 0A mbre?: .%.Tu fortuna es:. %.
000007df 00 01 1B 03 3B 30 00 00 00 05 00 00 00 50 FB FF FF 4C 00 00 00 AB FC FF FF 70 00 00 00 F0 FC ....;0.....P.....L.....p.....
000007fe FF FF 90 00 00 00 70 FD FF FF BC 00 00 00 D0 FD FF FF 08 01 00 00 14 00 00 00 00 00 00 00 01 .....p.....
0000081d 7A 52 00 01 7C 08 01 1B 0C 04 04 88 01 00 00 20 00 00 00 1C 00 00 00 FC FA FF FF 50 00 00 00 zR..|.....P...

```

Signed 8 bit:	127	Signed 32 bit:	2135247942	Hexadecimal:	7F 45 4C 46
Unsigned 8 bit:	127	Unsigned 32 bit:	2135247942	Decimal:	127 069 076 070
Signed 16 bit:	32581	Float 32 bit:	2,622539E+38	Octal:	177 105 114 106
Unsigned 16 bit:	32581	Float 64 bit:	1,16843158668567E+305	Binary:	01111111 01000101 01001100 01000110
<input type="checkbox"/> Show little endian decoding		<input type="checkbox"/> Show unsigned as hexadecimal		ASCII Text:	ELF
				Offset: 0x0 / 0x1d03	Selection: None INS

2. Haga más corto el mensaje de bienvenida.

```

00 74 75 20 77 69 6C 6C 20 66 6F 72 67 65 74 20 74 68 69 73 20 66 6F 72 65 64 20 66 ew me...Your society will be so
20 74 61 73 74 65 20 61 6E 64 20 72 65 66 ught by people of taste and ref
6C 20 62 65 20 68 6F 6E 6F 72 65 64 20 66 inement...You will be honored f
6F 75 72 20 74 69 6D 65 20 61 6E 64 20 73 or contributing your time and s
63 61 75 73 65 2E 00 45 78 70 65 63 74 20 kill to a worthy cause..Expect
68 65 20 6C 65 61 73 74 20 79 6F 75 20 63 the worst, it's the least you c
74 20 67 65 74 20 74 68 69 73 20 66 6F 72 an do..You may not get this for
20 61 20 61 6C 20 61 64 69 76 69 6E 61 64 tune...Bienvenido a al adivinad
00 45 73 74 65 20 6D 65 6E 73 61 6A 65 20 or de la fortuna! Este mensaje
75 65 64 65 20 73 65 72 20 6D 75 79 20 6D es muy largo, y puede ser muy m
2E 20 54 61 6C 20 76 65 7A 20 64 65 62 65 olesto al usuario. Tal vez debe
00 43 75 61 6C 20 65 73 20 74 75 20 6E 6F riamos acortarlo?.Cual es tu no

```

veamos que solo poniendo un "\0" que el bless lo ve como un punto pero si vemos el binario es un 00, con esto tiene que alcanzar

```

./2_1_fortune-recortado
Bienvenido a al adivinador de la fortuna!
Cual es tu nombre?: d

Tu fortuna es:
Never be led astray onto the path of virtue.

```

Ejercicio 3

La herramienta **objdump** de Linux permite hacer un análisis de archivos objeto revelando información importante de cómo está compuesto dicho archivo.

1. Haga un **disassembly** del código objeto y deduzca cómo es que se elige la fortuna a mostrarle al usuario. Si agrega el argumento “-M intel” podrá ver dicho código en formato Intel.

Ojito, hay que usar el -d para hacer disassembly

```
$ objdump -d -M intel 2_1_fortune-recortado
```

```
objdump - display information from object files
```

```

0804848b: <fortune>:
804848b: 55          push    ebp
804848c: 89 e5       mov     ebp,esp
804848e: 83 ec 10    sub     esp,0x10
8048491: c7 45 fc 00 00 00 00 mov     DWORD PTR [ebp-0x4],0x0
8048498: eb 08       jmp     80484a2 <fortune+0x17>
804849a: 83 45 fc 01 add     DWORD PTR [ebp-0x4],0x1
804849e: 83 45 08 01 add     DWORD PTR [ebp+0x8],0x1
80484a2: 8b 45 08     mov     eax,DWORD PTR [ebp+0x8]
80484a5: 0f b6 00    movzx   eax,BYTE PTR [eax]
80484a8: 84 c0       test    al,al
80484aa: 75 ee       jne     804849a <fortune+0xf>
80484ac: 8b 4d fc     mov     ecx,DWORD PTR [ebp-0x4]
80484af: ba ab aa aa aa mov     edx,0xaaaaaaaaab
80484b4: 89 c8       mov     eax,ecx
80484b6: f7 e2       mul     edx
80484b8: c1 ea 02    shr     edx,0x2
80484bb: 89 d0       mov     eax,edx
80484bd: 01 c0       add     eax,eax
80484bf: 01 d0       add     eax,edx
80484c1: 01 c0       add     eax,eax
80484c3: 29 c1       sub     ecx,eax
80484c5: 89 ca       mov     edx,ecx
80484c7: 8b 04 95 24 a0 04 08 mov     eax,DWORD PTR [edx*4+0x804a024]
80484ce: c9         leave
80484cf: c3         ret

```

Voy a tomar solo la seccion fortune que asumo que es la que calcula que string debe imprimir.

Lo pase a vsc para comentar

```

0804848b <fortune>:
804848b: 55          push    ebp
; ebp es el registro que mide la distancia relativa entre el sp y la siguiente instruccion
; me guardo eso en la pila para saber donde volver
804848c: 89 e5       mov     ebp,esp
; Ahora cargo en ebp la direccion del actual sp
804848e: 83 ec 10    sub     esp,0x10
; Seteo el sp 16 bytes más atras, de esta forma seteando un
; espacio de 16Bytes para el stack de la "funcion"
8048491: c7 45 fc 00 00 00 00 mov     DWORD PTR [ebp-0x4],0x0
; Voy 4Bytes atras del comienzo de mi pila y seteo un puntero a Double Word (16 bits / 2 Bytes) en 0
8048498: eb 08       jmp     salto1 ;80484a2 <fortune+0x17>
; Hago un salto incondicional a 80484a2 por claridad le voy a cambiar el nombre al rotulo
salto2: 83 45 fc 01 add     DWORD PTR [ebp-0x4],0x1
; en mi "variable" ya declarada acumulo 1
804849e: 83 45 08 01 add     DWORD PTR [ebp+0x8],0x1
; tambien le cargo 1 a ebp+8
salto1: 8b 45 08     mov     eax,DWORD PTR [ebp+0x8] ;80484a2 <fortune+0x17>
; Cargo en eax un puntero, ubicado 8Bytes adelante de mi pila, a Double Word (16 bits / 2 Bytes)
80484a5: 0f b6 00    movzx   eax,BYTE PTR [eax]
; Por lo que entiendo carga en eax lo que apunta eax ?
80484a8: 84 c0       test    al,al
; agarra la parte baja de eax y si es cero setea la ZF flag en 1
80484aa: 75 ee       jne     salto2 ;804849a <fortune+0xf>
; si ZF esta en 0 salta cambio el rotulo por comodidad
; entonces si la parte baja de eax no es cero salto
; si no entiendo mal es como un bucle hasta que termine algo null terminated? creo
80484ac: 8b 4d fc     mov     ecx,DWORD PTR [ebp-0x4]
; cargo en ecx la cantidad de veces que hice el "bucle"
80484af: ba ab aa aa aa mov     edx,0xaaaaaaaaab
; cargo un numero muy grande en edx
80484b4: 89 c8       mov     eax,ecx
; cargo en eax el contador
80484b6: f7 e2       mul     edx
; edx:eax=eax*edx
; se guarda la multiplicacion en 2 registros porque puede que de muy grande y no entre
80484b8: c1 ea 02    shr     edx,0x2
; es como dividir por 4 (desplaza los bits 2 bits a la derecha)
80484bb: 89 d0       mov     eax,edx
; cargo en eax edx
80484bd: 01 c0       add     eax,eax ; eax= 2*edx
80484bf: 01 d0       add     eax,edx ; eax= 3*edx
80484c1: 01 c0       add     eax,eax ; eax= 6*edx
80484c3: 29 c1       sub     ecx,eax
; a la cantidad de veces que hice el bucle le resto eax
80484c5: 89 ca       mov     edx,ecx
80484c7: 8b 04 95 24 a0 04 08 mov     eax,DWORD PTR [edx*4+0x804a024]
;?
80484ce: c9         leave
; Restaura el puntero base (ebp) y la pila.
80484cf: c3         ret
; vuelve

```

Reserva espacio para el stack, guarda la direccion de retorno y setea el sp de la funcion

Lee algo, asumo un string y cuenta cuantos caracteres tiene

Probando deduje que el string que lee es "tu nombre"

Realiza cuentas usando la longitud del string para generar un numero aleatorio

Retorna a donde fue llamada la seccion

veamos que pasa si cambio la longitud de mi nombre

<pre>Cual es tu nombre?: 1 Tu fortuna es: Never be led astray onto the path of virtue.</pre>	<pre>Cual es tu nombre?: a Tu fortuna es: Never be led astray onto the path of virtue.</pre>
--	--

```
Cual es tu nombre?: r
Tu fortuna es:
Never be led astray onto the path of virtue.
```

veamos que uso distintos nombre pero como la longitud es la misma la fortuna tambien

```
Cual es tu nombre?: 12
Tu fortuna es:
You will forget that you ever knew me.
```

```
Cual es tu nombre?: 123

Tu fortuna es:
Your society will be sought by people of taste and refinement.
```

```
Cual es tu nombre?: 1234

Tu fortuna es:
You will be honored for contributing your time and skill to a worthy cause.
```

```
Cual es tu nombre?: 12345

Tu fortuna es:
Expect the worst, it's the least you can do.
```

```
Cual es tu nombre?: 123456

Tu fortuna es:
Break into jail and claim police brutality.
```

```
Cual es tu nombre?: 1234567
Tu fortuna es:
Never be led astray onto the path of virtue.
```

```
Cual es tu nombre?: 12345678
Tu fortuna es:
You will forget that you ever knew me.
```

veamos que se empiezan a repetir, por lo tanto en algun lado esta haciendo moludo la cantidad de fortunas

2. Investigue cuales son las secciones que tiene dicho archivo. ¿En qué sección se encuentran los Strings?

```
.init
.plt
.plt.got
.text
.fini
```

-d
solo hace
disassembly de
algunas cosas

.interp	.eh_frame
.note.ABI-tag	.init_array
.note.gnu.build-id	.fini_array
.gnu.hash	.jcr
.dynsym	.dynamic
.dynstr	.got
.gnu.version	.got.plt
.gnu.version_r	.data
.rel.dyn	.comment
.rel.plt	
.rodata	
.eh_frame_hdr	

-D
disassembly de todo

objdump -s -j .rodata -M intel 2_1_fortune-recortado

veamos que si busco solo en .rodata (read only data) y le pido que me muestre la seccion en forma de bytes (-s) observamos lo siguiente

```
objdump -s -j .rodata -M intel 2_1_fortune-recortado
```

```
2_1_fortune-recortado:      formato del fichero elf32-i386
```

```
Contenido de la sección .rodata:
```

```
80485c8 03000000 01000200 42726561 6b20696e .....Break in
80485d8 746f206a 61696c20 616e6420 636c6169 to jail and clai
80485e8 6d20706f 6c696365 20627275 74616c69 m police brutali
80485f8 74792e00 4e657665 72206265 206c6564 ty..Never be led
8048608 20617374 72617920 6f6e746f 20746865 astray onto the
8048618 20706174 68206f66 20766972 7475652e path of virtue.
8048628 00000000 596f7520 77696c6c 20666f72 ....You will for
8048638 67657420 74686174 20796f75 20657665 get that you eve
8048648 72206b6e 6577206d 652e0000 596f7572 r knew me...Your
8048658 20736f63 69657479 2077696c 6c206265 society will be
8048668 20736f75 67687420 62792070 656f706c sought by peopl
8048678 65206f66 20746173 74652061 6e642072 e of taste and r
8048688 6566696e 656d656e 742e0000 596f7520 efinement...You
8048698 77696c6c 20626520 686f6e6f 72656420 will be honored
80486a8 666f7220 636f6e74 72696275 74696e67 for contributing
80486b8 20796f75 72207469 6d652061 6e642073 your time and s
80486c8 6b696c6c 20746f20 6120776f 72746879 kill to a worthy
80486d8 20636175 73652e00 45787065 63742074 cause..Expect t
80486e8 68652077 6f727374 2c206974 27732074 he worst, it's t
80486f8 6865206c 65617374 20796f75 2063616e he least you can
8048708 20646f2e 00596f75 206d6179 206e6f74 do..You may not
8048718 20676574 20746869 7320666f 7274756e get this fortun
8048728 65000000 4269656e 76656e69 646f2061 e...Bienvenido a
8048738 20616c20 61646976 696e6164 6f722064 al adivinador d
8048748 65206c61 20666f72 74756e61 21004573 e la fortuna!.Es
8048758 7465206d 656e7361 6a652065 73206d75 te mensaje es mu
8048768 79206c61 72676f2c 20792070 75656465 y largo, y puede
8048778 20736572 206d7579 206d6f6c 6573746f ser muy molesto
8048788 20616c20 75737561 72696f2e 2054616c al usuario. Tal
8048798 2076657a 20646562 65726961 6d6f7320 vez deberiamos
80487a8 61636f72 7461726c 6f3f0043 75616c20 acortarlo?.Cual
80487b8 65732074 75206e6f 6d627265 3f3a2000 es tu nombre?: .
```

3. Identifique todas las etiquetas del archivo. ¿Cuales reconoce?

<.interp>	<frame_dummy>
<.note.ABI-tag>	<fortune>
<.note.gnu.build-id>	<main>
<.gnu.hash>	<__libc_csu_init>
<.dynsym>	<__libc_csu_fini>
<.dynstr>	<_fini>
<.gnu.version>	<_fp_hw>
<.gnu.version_r>	<_IO_stdin_used>
<.rel.dyn>	<__GNU_EH_FRAME_HDR>
<.rel.plt>	<__FRAME_END__-0xe8>
<_init>	<__FRAME_END__>
<.plt>	<__frame_dummy_init_array_entry>
<printf@plt>	<__do_global_dtors_aux_fini_array_entry>
<puts@plt>	<__JCR_END__>
<__libc_start_main@plt>	<_DYNAMIC>
<__isoc99_scanf@plt>	<.got>
<__gmon_start__@plt>	<_GLOBAL_OFFSET_TABLE_>
<_start>	<__data_start>
<_x86.get_pc_thunk.bx>	<__dso_handle>
<deregister_tm_clones>	<fortunes>
<register_tm_clones>	<not_fortune>
<__do_global_dtors_aux>	<.comment>

Ejercicio 4

Con los conocimientos adquiridos en los puntos anteriores,

1. Obtenga la contraseña del programa password_easy.

password_easy

0000045c	00	85	D2	74	12	55	89	E5	83	EC	10	50	68	28	A0	04	08	FF	D2	83	C4	10	C9	F3	C3	8D	74	26	00	8D	BC	...	t.U.....Ph(.....t&...
0000047b	27	00	00	00	80	3D	28	A0	04	08	00	75	13	55	89	E5	83	EC	08	E8	7C	FF	FF	FF	C6	05	28	A0	04	08	'.....=(...u.U.....(...		
0000049a	01	C9	F3	C3	66	90	B8	10	9F	04	08	8B	10	85	D2	75	05	EB	93	8D	76	00	BA	00	00	00	00	85	D2	74	F2f.....u....v.....t.	
000004b9	55	89	E5	83	EC	14	50	FF	D2	83	C4	10	C9	E9	75	FF	FF	FF	8D	4C	24	04	83	E4	F0	FF	71	FC	55	89	E5	U.....P.....u....L\$.....q.U..	
000004d8	51	83	EC	24	83	EC	0C	68	D0	85	04	08	E8	97	FE	FF	FF	83	C4	10	83	EC	08	8D	45	DA	50	68	E7	85	04	Q..\$....h.....E.Ph...	
000004f7	08	E8	B3	FE	FF	FF	83	C4	10	83	EC	08	8D	45	DA	50	68	EA	85	04	08	E8	5F	FE	FF	FF	83	C4	10	85	C0E.Ph.....	
00000516	75	12	83	EC	0C	68	F4	85	04	08	E8	6B	FE	FF	FF	83	C4	10	EB	10	83	EC	0C	68	08	86	04	08	E8	59	FE	u....h.....k.....h.....Y.	
00000535	FF	FF	83	C4	10	B8	00	00	00	00	8B	4D	FC	C9	8D	61	FC	C3	66	90	66	90	66	90	90	55	57	56	53M...a..f.f.f.f..UWVS			
00000554	E8	A7	FE	FF	FF	81	C3	A7	1A	00	00	83	EC	0C	8B	6C	24	20	8D	B3	0C	FF	FF	FF	E8	BF	FD	FF	FF	8D	83l\$	
00000573	08	FF	FF	FF	29	C6	C1	FE	02	85	F6	74	25	31	FF	8D	B6	00	00	00	00	83	EC	04	FF	74	24	2C	FF	74	24).....t%1.....t\$,t\$	
00000592	2C	55	FF	94	BB	08	FF	FF	FF	83	C7	01	83	C4	10	39	FE	75	E3	83	C4	0C	5B	5E	5F	5D	C3	8D	76	00	F3	,U.....9.u....[^_].v..	
000005b1	C3	00	00	53	83	EC	08	E8	43	FE	FF	81	C3	43	1A	00	00	83	C4	08	5B	C3	03	00	00	00	01	00	02	00	...	S....C....C.....[.....	
000005d0	49	6E	67	72	65	73	65	20	6C	61	20	43	6F	6E	74	72	61	73	65	6E	61	3A	00	25	73	00	53	55	43	55	54	Ingrese la Contraseña:.%s.SUCUT	
000005ef	52	55	4C	45	00	43	6F	6E	74	72	61	73	65	6E	61	20	43	6F	72	72	65	63	74	61	00	43	6F	6E	74	72	61	RULE.Contrasena Correcta.Contra	
0000060e	73	65	6E	61	20	49	6E	63	6F	72	72	65	63	74	61	00	00	00	01	1B	03	3B	28	00	00	04	00	00	00	40	00	sena Incorrecta.....;(.....@	
0000062d	FD	FF	FF	44	00	00	00	AB	FE	FF	FF	68	00	00	00	30	FF	FF	FF	94	00	00	00	90	FF	FF	FF	E0	00	00	00	...D.....h...0.....	
0000064c	14	00	00	00	00	00	00	01	7A	52	00	01	7C	08	01	1B	0C	04	04	88	01	00	00	20	00	00	00	1C	00	00	00zR..	
000006b1	00	F4	FC	FF	FF	60	00	00	00	00	0E	08	46	0E	0C	4A	0F	0B	74	04	78	00	3F	1A	3B	2A	32	24	22	28	00F..J..t.x.?.;*2\$"(.	
0000068a	00	00	40	00	00	00	3B	FE	FF	FF	7C	00	00	00	00	44	0C	01	00	47	10	05	02	75	00	43	0F	03	75	7C	06	..@...;... ...D...G...u.C...u .	
000006a9	02	69	0C	01	00	41	C5	43	0C	04	04	48	00	00	00	6C	00	00	00	94	FE	FF	FF	5D	00	00	00	00	41	0E	08	.i...A.C...H...l.....]....A.	

vamos al bless y encontramos un string SUCUTRULE, veamos si es la contraseña

```
nahuel-prado@nahuel-prado-HP-245-G8-Notebook-PC:~/Escritorio/Estudio/Arqui/Tp1/binarios$ ./password_easy
Ingrese la Contraseña:SUCUTRULE
Contraseña Correcta
```

2. Cámbiela por "1234".

volvemos al bless y cambiamos la contraseña

F 74 24).....t%1.....t\$,t\$
6 00 F3	,U.....9.u....[^_].v..
0 02 00	...S....C....C.....[.....
3 34 00	Ingrese la Contraseña:.%s.1234.
4 72 61	RULE.Contrasena Correcta.Contra
0 00 40	sena Incorrecta.....;(.....@
0 00 00	...D.....h...0.....
C 00 00zR..

Probamos y funciona :)

```
nahuel-prado@nahuel-prado-HP-245-G8-Notebook-PC:~/Escritorio/Estudio/Arqui/Tp1$ ./4_2_password_easy-hacked
Ingrese la Contraseña:1234
Contraseña Correcta
```

Ejercicio 5

Con los conocimientos adquiridos en los puntos anteriores,

1. Obtenga la contraseña del programa password_ofuscated1

Pareciera que el bless esta vez no va a ser de mucha ayuda

Usamos el siguiente comando
objdump -d -M intel binarios/password_ofuscated
y vemos que etiquetas tiene, si vamos a main veremos que en un momento se llama a fillpassword
vamos a revisar que hace esta etiqueta.

```

080484cb <fillpassword>:
80484cb: 55                push    ebp
80484cc: 89 e5            mov     ebp,esp

80484ce: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
;en eax se carga el primer valor de la pila
80484d1: c6 00 53         mov     BYTE PTR [eax],0x53
;resulta que era un puntero en el cual se carga 0x53 (S)
80484d4: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
;no afecta?
80484d7: 83 c0 01         add     eax,0x1
;eax se mueve a la siguiente posision
80484da: c6 00 30         mov     BYTE PTR [eax],0x30
;le cargo 0x30 (0)
80484dd: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
80484e0: 83 c0 02         add     eax,0x2
;avanzo un lugar en el "vector"
80484e3: c6 00 52         mov     BYTE PTR [eax],0x52
;cargo 0x52 (R)
80484e6: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
80484e9: 83 c0 03         add     eax,0x3
;avanzo un lugar en el "vector"
80484ec: c6 00 50         mov     BYTE PTR [eax],0x50
;cargo 0x50 (P)
80484ef: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
80484f2: 83 c0 04         add     eax,0x4
;avanzo un lugar en el "vector"
80484f5: c6 00 52         mov     BYTE PTR [eax],0x52
;cargo 0x52 (R)
80484f8: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
80484fb: 83 c0 05         add     eax,0x5
;avanzo un lugar en el "vector"
80484fe: c6 00 33         mov     BYTE PTR [eax],0x33
;cargo 0x33 (3)
8048501: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
8048504: 83 c0 06         add     eax,0x6
;avanzo un lugar en el "vector"
8048507: c6 00 53         mov     BYTE PTR [eax],0x53
;cargo 0x53 (S)
804850a: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
804850d: 83 c0 07         add     eax,0x7
;avanzo un lugar en el "vector"
8048510: c6 00 34         mov     BYTE PTR [eax],0x34
;cargo 0x34 (4)
8048513: 8b 45 08         mov     eax,DWORD PTR [ebp+0x8]
8048516: 83 c0 08         add     eax,0x8
;avanzo un lugar en el "vector"
8048519: c6 00 00         mov     BYTE PTR [eax],0x0
;en ultima posision pone un 0
804851c: 90                nop
804851d: 5d                pop     ebp
804851e: c3                ret

```

veamos que esta funcion genera un String "SORPR3S4\0"
probemos si es la contraseña

:)

```

nahuel-prado@nahuel-prado-HP-245-G8-Notebook-PC:~/Escritorio/Estudio/Arqui/Tp1$ ./binarios/password_ofuscated
Ingrese la Contraseña:S0RPR3S4
Contraseña Correcta

```

2. Cámbiela por "1234"

Bien costo bastante pero aquí la solución:

Primero debemos encontrar donde esta la "funcion" fillpassword en bless, para esto usaremos el "binario" (el hexa) dado por objdump luego con la lupa del bless encontramos donde esta esta la etiqueta. Finalmente siguiendo un poco el codigo y otro poco el hexa vemos cuales son los caracteres a modificar. Logrando que el generador de contraseñas nos de 1234 y no S0RPR3S4.

```

U.....P.....u...U...E...1.E..
.....2.E.....3.E.....4.E.....
.E.....3.E.....S.E.....4.E..
.....1...I,$.....a.U...O...D.E.P.

```

```

55 89 E5 83 EC 14 50 FF D2 83 C4 10 C9 E9 75 FF FF FF 55 89 E5 8B 45 08 C6 00 31 8B 45 08 83
C0 01 C6 00 32 8B 45 08 83 C0 02 C6 00 33 8B 45 08 83 C0 03 C6 00 34 8B 45 08 83 C0 04 C6 00
00 8B 45 08 83 C0 05 C6 00 33 8B 45 08 83 C0 06 C6 00 53 8B 45 08 83 C0 07 C6 00 34 8B 45 08

```

```

080484cb <fillpassword>:
80484cb: 55          push    ebp
80484cc: 89 e5      mov     ebp,esp
80484ce: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
80484d1: c6 00 53  mov     BYTE PTR [eax],0x53
80484d4: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
80484d7: 83 c0 01   add     eax,0x1
80484da: c6 00 30  mov     BYTE PTR [eax],0x30
80484dd: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
80484e0: 83 c0 02   add     eax,0x2
80484e3: c6 00 52  mov     BYTE PTR [eax],0x52
80484e6: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
80484e9: 83 c0 03   add     eax,0x3
80484ec: c6 00 50  mov     BYTE PTR [eax],0x50
80484ef: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
80484f2: 83 c0 04   add     eax,0x4
80484f5: c6 00 52  mov     BYTE PTR [eax],0x52
80484f8: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
80484fb: 83 c0 05   add     eax,0x5
80484fe: c6 00 33  mov     BYTE PTR [eax],0x33
8048501: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
8048504: 83 c0 06   add     eax,0x6
8048507: c6 00 53  mov     BYTE PTR [eax],0x53
804850a: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
804850d: 83 c0 07   add     eax,0x7
8048510: c6 00 34  mov     BYTE PTR [eax],0x34
8048513: 8b 45 08   mov     eax,DWORD PTR [ebp+0x8]
8048516: 83 c0 08   add     eax,0x8
8048519: c6 00 00  mov     BYTE PTR [eax],0x0
804851c: 90          nop

```


804851d:	5d	pop	ebp
804851e:	c3	ret	

Ejercicio 6

Considere el siguiente programa:

```
#include <stdio.h>

#define DIMENSION 1024

int matriz[DIMENSION][DIMENSION];

int main(void) {
    printf("Hello World!\n");
    return 0;
}
```

Compile y analice el tamaño del binario final. Piense, ¿Cuánto espacio ocupa una matriz de 1024x1024 de ints? ¿Dónde está toda esa información en el binario? ¿Qué tamaño esperaba?

¿Qué ocurriría si dicha matriz estuviera inicializada con 0's? ¿Y con otro valor? Pruebe cómo varía el tamaño de dicho binario con las distintas alternativas.

```
int matriz[DIMENSION][DIMENSION] = {0};
```

una matriz de 1024x1024 ints debería ocupar 4×1024^2 Bytes? asumo que debería estar en .data o .rodata 248bytes pesa, si solo la cargo con 0 no cambia, ahora si la cargo con {{1,0}} refleja mejor el numero que habia afirmado.

Ejercicio 7

Utilice el programa **Evan's Debugger** con el programa **fortune** de los puntos anteriores. Corra cualquier programa e identifique los siguientes elementos de cada programa:

- Zona de Código
- Zona de Datos
- Stack

The screenshot shows the Evan's Debugger interface with the following components:

- Registers:** Shows the state of various registers, including EAX, ECX, EDX, EBX, ESP, EBP, ESI, and EDI.
- Data Dump:** Shows a memory dump starting at address 0x08048000, displaying hexadecimal and ASCII values.
- Stack:** Shows the current stack frame, including the return address and arguments.
- Debugger Error Console:** Displays the output of the program, including the terminal output and the program's exit status.

- ¿En qué lugar físico de la PC está la información que está visualizando?
- ¿Algunas secciones están solapadas?
- ¿Por qué en cada pantalla la información visualizada es distinta?
- ¿Qué diferencia hay con los ejemplos anteriores?

2. Creo que directamente no se ven las secciones separadas

3. En realidad es la misma informacion pero bajo diferentes interpretaciones

4. Asumo que se refiere al objdump, strings y el bless, la diferencia esta en que los cambios se ven en tiempo real y que es un rejunte de todo en uno.

- ¿En qué parte de la PC está la información que está analizando?

[illegible]

2. Confirme su suposición de cómo se está calculando la fortuna del usuario.

ni idea

3. Abra el Programa **password_ofuscated1** y observe paso a paso cómo cambia la sección de datos a medida que se va generando la contraseña. mirar el stack

4. Modifique y guarde dicho programa de tal forma que sin importar la contraseña, siempre de cómo correcta.

voy al main y modifico el jne a jmp y luego hago que el salto sea a 0 es decir siguiente

Esta es la linea original sin modificar

eb significa jmp y 10 significa 10 bytes de desplazamiento

```
./7_2password_ofuscated-super-hacked qw
Ingrese la Contraseña:j
Contraseña Correcta
nahuel-prado@nahuel-prado-HP-245-G8-Note
./7_2password_ofuscated-super-hacked qw
Ingrese la Contraseña:dqsdfas
Contraseña Correcta
nahuel-prado@nahuel-prado-HP-245-G8-Note
./7_2password_ofuscated-super-hacked
Ingrese la Contraseña:2wy4wdrhufadxzv
Contraseña Correcta
```

Ejercicio 8

Utilice el **debugger** para deducir cómo son las contraseñas que son válidas para el programa **password_hard**.

```
00484cb: <check_password>:
80484cb: 55          push    ebp
80484cc: 89 e5       mov     ebp,esp
80484ce: 83 ec 18    sub     esp,0x18
80484d1: c7 45 f0 01 00 00 00 mov     DWORD PTR [ebp-0x10],0x1

; Llamada a strlen para saber la long de la contraseña ingresada

80484d8: 83 ec 0c    sub     esp,0xc
80484db: ff 75 08    push   DWORD PTR [ebp+0x8]
80484de: e8 ad fe ff ff call    8048390 <strlen@plt>
80484e3: 83 c4 10    add     esp,0x10
80484e6: 89 45 ec    mov     DWORD PTR [ebp-0x14],eax

; Si la long.eql(8) bien, sino a fuera

80484e9: 83 7d ec 08 cmp     DWORD PTR [ebp-0x14],0x8
80484ed: 74 07       je      80484f6 <check_password+0x2b> ;si son iguales skip

80484ef: b8 00 00 00 00 mov     eax,0x0 ;carga el 0 para hacer el return
80484f4: eb 34       jmp     804852a <check_password+0x5f> ;return 0

80484f6: c6 45 f7 00 mov     BYTE PTR [ebp-0x9],0x0 ;inicializa en 0
80484fa: eb 1f       jmp     804851b <check_password+0x50>

80484fc: 8b 45 08    mov     eax,DWORD PTR [ebp+0x8];Carga el puntero a la cadena de la contraseña en eax
80484ff: 0f b6 00    movzx   eax,BYTE PTR [eax];Carga el primer carácter de la contraseña en eax
8048502: 3a 45 f7    cmp     al,BYTE PTR [ebp-0x9];Compara este carácter con el carácter previo almacenado
8048505: 7d 07       jge     804850e <check_password+0x43>;Si el carácter actual es mayor o igual al anterior, hace el salto

8048507: b8 00 00 00 00 mov     eax,0x0;el caracter era menor q el anterior
804850c: eb 1c       jmp     804852a <check_password+0x5f> ;return 0

804850e: 8b 45 08    mov     eax,DWORD PTR [ebp+0x8]
8048511: 0f b6 00    movzx   eax,BYTE PTR [eax];Carga el carácter actual
8048514: 88 45 f7    mov     BYTE PTR [ebp-0x9],al;Guarda este carácter en la variable temporal
8048517: 83 45 08 01 add     DWORD PTR [ebp+0x8],0x1;Avanza al siguiente carecter

804851b: 8b 45 08    mov     eax,DWORD PTR [ebp+0x8];
804851e: 0f b6 00    movzx   eax,BYTE PTR [eax]
8048521: 84 c0       test    al,al ;verifica si es NULL
8048523: 75 d7       jne     80484fc <check_password+0x31>
8048525: b8 01 00 00 00 mov     eax,0x1
804852a: c9         leave   eax
804852b: c3         ret
```

long == 8

Verifica si la contraseña es de longitud 8 y luego que sus caracteres sean ordenados de menor a mayor si es así, entonces es una contraseña valida (retorna 1)