Technische Informatik 2
Tutor: Marc Hildebrandt

WS 2016/17

C08

Timo Jasper (Inf, 3.FS.)
Thomas Tannous (Inf, 3.FS.)
Oliver Hilbrecht (Inf, 3.FS.)
Moritz Gerken (Inf, 3.FS.)

Übungsblatt 2

1	2	3	4	5	$\sum$

Lösungsvorschlag
Abgabe: 14.11.2016

1 Aufgabe 1

Hier haben wir mit den Befehlen cat, tail und cut die nötigen Stellen aus dem Tag rausgeschnitten, die wir benötigten. Dabei ist der String leer, falls nichts an der Stelle keine Angabe war. So haben wir wie erwartet die Angabe durch ein _ ersetzt. Für die Titelnummer haben wir den Befehl dd benutzt um die nötigen Bytes zu bekommen. Mit hexdump -e .. kriegen wir dann die Tracknumme aus dem Binärcode in eine Dezimal konvertiert.

```
#!/bin/bash
#ausf hrbar machen: chmod +x /pfad/zu/mp3-rename.sh

#TODO greift irgendwie noch nicht (richtig) auf den tag zu
#TODO wenn tag datei nicht beschrieben kommen h ssliche Err.



#benennt datei des   bergebenen   Pfades korrekt um
myrename (){
      pathfile=$1

      kuenstler=$(cat "$1" | tail -c 128 | cut -b 33-62)
      if [ "$kuenstler" = "" ]
      then
          kuenstler='_'
      fi

      album=$(cat "$1" | tail -c 128 | cut -b 63-92)
      if [ "$album" = "" ]
      then
          album='_'
      fi

      titel=$(cat "$1" | tail -c 128 | cut -b 4-33)
      if [ "$titel" = "" ]
      then
          titel='_'
      fi

     NN=$(echo $((16#$(tail -c 128 $1 | dd skip=126 count=1 ibs=1
         status=none | hexdump -e'1/1 "%02x"' -e'"\n"'))))
```

```
        if [ $NN -lt 10 ]
        then NN="-0"$NN
        else
            NN="-"$NN
        fi
        newname=("$kuenstler""-""$album""-""$titel""$NN"".mp3")
        dir=$(dirname $pathfile)"/"
        pathtofilename=$dir$newname
        mv $1 "$pathtofilename"
        echo -e "rename $pathfile to $pathtofilename"
}

for var in "$@"
do
    myrename $var
done
```

1.1 Tests

Unsere Tests prüfen auf die wichtigen Randfälle.

```
#!/bin/bash

mkdir mp3s
cp *.mp3 mp3s/
cp the\ programmer/*.mp3 mp3s/
cp unknown mp3s

./mp3-rename.sh mp3s/*

#nice coloring
RED='\033[0;31m'
GREEN='\033[0;32m'
NC='\033[0m'
succ="${GREEN}SUCCESS${NC}"
fail="${RED}FAILED${NC}"

testprint () {

    if [ "$1" = "$2" ]
    then
        echo -e "        $succ"
    else
        echo -e "        ${fail}" >&2
        echo -e "$3 ist nicht der richtige Name"
    fi
}

## declare an array variable
```

```
declare -a correctfnarr=("BGP-Distance Vectors-RIP-07.mp3"
                         "_-Bugs galore-Endless Recursion-01.mp3"
                         "_-Bugs galore-Race Condition-23.mp3"
                         "_-Bugs galore-Three little bugs-112.mp3"
                         "J. Postel & The Packet Drops-Source Quench-
                            Fragments of IP-01.mp3"
                         "J. Postel & The Packet Drops-Source Quench-
                            No Route to Host-03.mp3"
                         "J. Postel & The Packet Drops-Source Quench-
                            Traffic Class Assignment-02.mp3"
                         "The Drammer Boys-_-_-36.mp3"
                         "The OpenSSH-Roaming Around-B.E.A.S.T.-05.
                            mp3"
                         "The OpenSSH-Roaming Around-Heartbleed-06.
                            mp3"
                         "The OpenSSH-Roaming Around-Triple 7-04.mp3"
)


for file in mp3s/*; do
    name="$(basename "$file")"
    onematch=false
    for str in "${correctfnarr[@]}"; do
        if [ "$str" = "$name" ]

            then
                testprint "$name" "$str"
                onematch=true
        fi
    done
    if [ $onematch = false ]
        then
            testprint "ochne" "nichtgleich" "${name}"
    fi
done

rm -rf mp3s/
```

2 Aufgabe 2

2.1 main

2.1.1 a (Text/Data-Relocation-Table, Symboltabelle und Stringtabelle für das a.out-Format)

Textsegment:

0	pushq %rbp
4	movq %rsp, %rbp
8	subq $32, %rsp
12	movl %edi, -20(%rbp)
16	movq %rsi, -32(%rbp)
20	cmpl $1, -20(%rbp)
24	jle +72
28	movq -32(%rbp), %rax
32	addq $8, %rax
36	movq (%rax), %rax
40	movl $10, %edx
44	movl $0, %esi
48	movq %rax, %rdi
52	call 0
56	movq %rax, -16(%rbp)
60	movq $0, -8(%rbp)
64	movq -8(%rbp), %rax
68	cmpq -16(%rbp), %rax
72	ja +32
76	movq -8(%rbp), %rax
80	movq %rax, %rdi
84	call 0
88	movq %rax, %rdi
92	call show__Fm
96	addq $1, -8(%rbp)
100	jmp -36
104	movq 0(%rip), %rax
108	movq %rax, %rsi
112	leaq cout(%rip), %rdi
116	call 0
120	movl $0, %eax
124	ret

Datensegment, Relocation Tabelle, SymbolTabelle, String Tabelle :

	(Datensegment ist leer)		
	52	1	
	84	2	
	92	3	
	116	4	
0	4	Text	0
1	9	UNDEFINED	
2	23	UNDEFINED	
3	31	UNDEFINED	
4	40	UNDEFINED	
0	Länge (4 Bytes)		
4	main		
8	\0str		
12	tol_		
16	_Fxc		
20	ci\0f		
24	ib__		
28	Fm\0s		
32	how_		
36	_Fm\0		
40	__ls		
44	__7o		
48	stre		
52	amm\0		

2.2 show

2.2.1 a (Text/Data-Relocation-Table, Symboltabelle und Stringtabelle für das a.out-Format)

0			pushq %rbp
4			movq %rsp, %rbp
8			subq $16, %rsp
12			movq %rdi, -8(%rbp)
16			movq -8(%rbp), %rax
20			movq %rax, %rsi
24			leaq cout(%rip), %rdi
28			call 0
32			movl $32, %esi
36			movq %rax, %rdi
40			movq %rax, %rdi
44			call 0
48			ret
	(Datensegment ist leer)		
	28	1	
	44	2	
0	4	Text	0
1	13	UNDEFINED	
2	29	UNDEFINED	
0	Länge (4 Bytes)		
4		show	
8		__Fm	
12		\0__l	
16		s__7	
20		ostr	
24		eamm	
28		\0__l	
32		s__7	
36		ostr	
40		eamc	
44		\	

2.3 fib

2.3.1 a (Text/Data-Relocation-Table, Symboltabelle und Stringtabelle für das a.out-Format)

0	pushq %rbp
4	movq %rsp, %rbp
8	pushq %rbx
12	subq $40, %rsp
16	movq %rdi, -40(%rbp)
20	movq $1, -24(%rbp)
24	cmpq $1, -40(%rbp)
28	jbe +48
32	movq -40(%rbp), %rax
36	subq $2, %rax
40	movq %rax, %rdi
44	call -44
48	movq %rax, %rbx
52	movq -40(%rbp), %rax
56	subq $1, %rax
60	movq %rax, %rdi
64	call -64
68	addq %rbx, %rax
72	movq %rax, -24(%rbp)
76	movq -24(%rbp), %rax
80	addq $40, %rsp
84	popq %rbx
88	popq %rbp
92	ret
	(Datensegment ist leer)
	(ext/Data Rel. Tab ist leer)
0	4 Text 0
0	Länge (4 Bytes)
4	fib_
8	_Fm\0

2.4 b

2.5 c

3 Aufgabe 3

Da man den Wert der Variable my_password nicht direkt auslesen kann, weil sie öptimized outïst (was so viel heißt wie der Code wurde optimiert und der Wert ist nicht einfach zugänglich), mussten wir einen anderen Weg wählen. Zuerst haben wir nach der Funktion geschaut die unser Passwort auf Gleichheit prüft. Dies war die check_password(char const*) Funktion. Anschließend haben wir sie disassembled. Dort haben dann nach dem vergleich geguckt und die Addresse des Befehls notiert. An der stelle haben wir dann ein Breakpoint gesetzt und das Programm laufen

lassen. Als gebreakt wurde, schauten wir was im Register $rsi nach einem String mit dem Befehl x/s $rsi. Dort befand sich dann zum Glück das Passwort "TeI2-ist_einfach".

GDB weiß so viel weil: Der Maschinencode kann zurück in assembler übersetzt werden. Die binary Datei selbst hat einen Teil der sich das Datensegment nennt, wo alle initialisierten Daten vorhanden sind. So kann gdb den Maschinencode verstehen, einen breakpoint setzen und Register des Prozessors auslesen.

4 Weitere Aufgaben