```
     1                                  ; taken from here cause I was banging my head against
a wall for hours -> https://gist.github.com/BertrandBordage/10921263
     2                                  ; checking if file exists -> https://gist.github.com/A
rchenoth/5380671
     3                                  %define SYS_EXIT 60
     4                                  %define SYS_READ 0
     5                                  %define SYS_WRITE 1
     6                                  %define SYS_OPEN 2
     7                                  %define SYS_CLOSE 3
     8                                  %define STDOUT 1
     9                                  %define SYS_CREATE 85
    10
    11                                  %define BUFFER_SIZE 180
    12
    13                                  section .text
    14                                  global  _start
    15                                  _start:
    16                                    ; So we can read in our argument from argv[]
    17 00000000 4883C410                  add rsp, byte 0x10
    18 00000004 5F                        pop rdi
    19 00000005 EB00                      jmp _check
    20                                    ; I added this to see
    21                                  _check:
    22                                  ; basic if/else control flow -> https://stackoverflow.
com/questions/14292903/complex-if-statement-in-assembly
    23 00000007 BA00000000                mov rdx,0
    24 0000000C 4839C2                    cmp rdx,rax
    25 0000000F 7E02                      jle _cont
    26 00000011 7F3C                      jnle _exit_failiure
    27                                    _cont:
    28
    29                                    ; open the file
    30 00000013 B802000000                mov rax, SYS_OPEN
    31 00000018 BE00000000                mov rsi, 0
    32 0000001D 0F05                      syscall
    33 0000001F 48890425[00000000]        mov [fd], rax
    34 00000027 EB00                      jmp _read_write
    35
    36                                  _read_write:
    37                                    ; Read the file into the buffer
    38 00000029 B800000000                mov rax, SYS_READ
    39 0000002E 488B3C25[00000000]        mov rdi, [fd]
    40 00000036 48BE-                     mov rsi, file_buffer
    41 00000038 [0000000000000000]
    42 00000040 BAB4000000                mov rdx, BUFFER_SIZE
    43 00000045 0F05                      syscall
    44
    45 00000047 4883F800                  cmp rax, 0
    46 0000004B 740E                      je _exit_success
    47
    48 0000004D 7ADA                      jp _read_write
    49
    50
    51                                  _exit_failiure:
    52                                    ; exit with code 1
    53 0000004F B83C000000                mov rax, 60
    54 00000054 BF01000000                mov rdi, 1
    55 00000059 0F05                      syscall
    56
    57                                  _exit_success:
    58                                    ; Close the file stream
    59 0000005B B803000000                mov rax, SYS_CLOSE
    60 00000060 48BF-                     mov rdi, fd
    61 00000062 [0000000000000000]
    62 0000006A 0F05                      syscall
```

```
63 0000006C EB00                    jmp exit
64                                  exit:
65 0000006E B801000000              mov rax, SYS_WRITE
66 00000073 BF01000000              mov rdi, 1
67 00000078 48BE-                   mov rsi, file_buffer
68 0000007A [0000000000000000]
69 00000082 BAB4000000              mov rdx, BUFFER_SIZE
70 00000087 0F05                    syscall
71 00000089 B801000000              mov rax, SYS_WRITE
72 0000008E BE0A000000              mov rsi, 10
73 00000093 BA01000000              mov rdx, 1
74 00000098 0F05                    syscall
75 0000009A B83C000000              mov rax, 60
76 0000009F BFB4000000              mov rdi, BUFFER_SIZE
77 000000A4 0F05                    syscall
78
79
80                                  section .data
81 00000000 0000                    fd dw 0
82 00000002 32361392A55A27F3        key db 0x32,0x36,0x13,0x92,0xa5,0x5a,0x27,0xf3
83 0000000A 646563727970746564      path: db "decrypted"
84 00000013 0000                    decrypted dw 0
85                                  led: equ $-path
86 00000015 01                      index db 1
87
88
89                                  section .bss
90 00000000 <res 000000B4>          file_buffer resb BUFFER_SIZE
91                                  length: equ $-file_buffer
92
93                                  ; key: db '0123456789ABCDEF', 10
```