```
     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                                  ; This program will decrypt a packet from an incoming
satallite
     4                                  ; It decrypts the packet in memory so there is no need
 to write subroutines to reverse the order of the bytes
     5                                  ; Written by Jared Dyreson
     6                                  ; CPSC-240 TR @ 11:30 to 13:20
     7
     8                                  %define SYS_EXIT 60
     9                                  %define SYS_READ 0
    10                                  %define SYS_WRITE 1
    11                                  %define SYS_OPEN 2
    12                                  %define SYS_CLOSE 3
    13                                  %define STDOUT 1
    14                                  %define SYS_CREATE 85
    15
    16                                  %define BUFFER_SIZE 180
    17
    18                                  section .text
    19                                  global  _start
    20                                  _start:
    21                                    ; So we can read in our argument from argv[]
    22 00000000 4883C410                  add rsp, byte 0x10
    23 00000004 5F                        pop rdi
    24 00000005 EB00                      jmp _check
    25                                  _check:
    26                                  ; basic if/else control flow -> https://stackoverflow.
com/questions/14292903/complex-if-statement-in-assembly
    27 00000007 BA00000000                mov rdx,0
    28 0000000C 4839C2                    cmp rdx,rax
    29 0000000F 7E02                      jle _cont
    30 00000011 7F3C                      jnle _exit_failiure
    31                                    _cont:
    32
    33                                    ; open the file
    34 00000013 B802000000                mov rax, SYS_OPEN
    35 00000018 BE00000000                mov rsi, 0
    36 0000001D 0F05                      syscall
    37 0000001F 48890425[00000000]        mov [fd], rax
    38 00000027 EB00                      jmp _read_write
    39
    40                                  _read_write:
    41                                    ; Read the file into the buffer
    42 00000029 B800000000                mov rax, SYS_READ
    43 0000002E 488B3C25[00000000]        mov rdi, [fd]
    44 00000036 48BE-                     mov rsi, file_buffer
    45 00000038 [0000000000000000]
    46 00000040 BAB4000000                mov rdx, BUFFER_SIZE
    47 00000045 0F05                      syscall
    48
    49 00000047 4883F800                  cmp rax, 0
    50 0000004B 740E                      je close_file
    51
    52 0000004D 7ADA                      jp _read_write
    53
    54
    55                                  _exit_failiure:
    56                                    ; exit with code 1
    57 0000004F B83C000000                mov rax, 60
    58 00000054 BF01000000                mov rdi, 1
    59 00000059 0F05                      syscall
    60
```

```
    61                                  close_file:
    62                                    ; Close the file stream
    63 0000005B B803000000                mov rax, SYS_CLOSE
    64 00000060 48BF-                     mov rdi, fd
    65 00000062 [0000000000000000]
    66 0000006A 0F05                      syscall
    67
    68 0000006C 4D31C0                    xor r8, r8
    69 0000006F 4831C0                    xor rax, rax
    70
    71 00000072 EB05                      jmp decryptor
    72                                  _reset_key:
    73 00000074 4831C0                    xor rax, rax
    74 00000077 EB00                      jmp decryptor
    75                                  decryptor:
    76                                    ; goal
    77                                    ; xor each byte in the file buffer, given a certain
offset to that position in the string in the file_buffer and the key
    78                                    ; reset the key once we reach 9th element
    79
    80                                    ; r8 -> indexing the file_buffer
    81                                    ; rax -> indexing our key
    82                                    ; r11 -> contains our needed variable from the file_
buffer
    83                                    ; r12 -> contains our needed variable from the key
    84
    85                                    ; if(r8 >= 180), we need to leave
    86 00000079 4981F8B4000000            cmp r8, 180
    87 00000080 7D42                      jge exit
    88
    89                                    ; if(r9 > 8), we need to reset it
    90 00000082 4883F808                  cmp rax, 8
    91 00000086 7FEC                      jg _reset_key
    92
    93                                    ; load the character from the file_buffer we need in
to the variable [check]
    94 00000088 488D1C25[00000000]        lea rbx, [file_buffer]
    95 00000090 4E8B1C03                  mov r11, [rbx+(r8*1)] ; variable = buff[i]
    96
    97
    98                                    ; move current offset into the correct register
    99 00000094 488D1C25[02000000]        lea rbx, [key]
   100
   101 0000009C 4883F800                  cmp rax, 0
   102 000000A0 7402                      je other
   103 000000A2 7508                      jne _begin
   104                                    other:
   105 000000A4 41BC36000000              mov r12, 0x36 ; 0th index cannot be accessed for som
e reason
   106 000000AA EB06                      jmp cont
   107                                    _begin:
   108 000000AC 4C8B24C3                  mov r12, [rbx+(rax*8)]; xor_key_variable = key[index
]
   109 000000B0 EB00                      jmp cont
   110                                    cont:
   111 000000B2 4D31E3                    xor r11, r12 ; buf[i] ^ key[index]
   112 000000B5 4D8998[00000000]          mov [file_buffer+r8], r11 ; r11 = buf[i] ^ key[index
]
   113 000000BC 49FFC0                    inc r8 ; r8++
   114 000000BF 48FFC0                    inc rax ; rax++
   115 000000C2 EBB5                      jmp decryptor ;loop back
   116
   117                                  exit:
   118                                    print_buffer:
   119 000000C4 48BE-                     mov rsi, file_buffer
```

```
120 000000C6 [0000000000000000]
121 000000CE B801000000                mov rax, SYS_WRITE
122 000000D3 BAB4000000                mov rdx, BUFFER_SIZE
123 000000D8 BF01000000                mov rdi, 1
124 000000DD 0F05                      syscall
125 000000DF B801000000                mov rax, SYS_WRITE
126 000000E4 48BE-                     mov rsi, endl
127 000000E6 [4A00000000000000]
128 000000EE BF01000000                mov rdi, 1
129 000000F3 BA01000000                mov rdx, 1
130 000000F8 0F05                      syscall
131 000000FA EB00                      jmp leave_segment
132                                    leave_segment:
133 000000FC B83C000000                mov rax, 60
134 00000101 BFB4000000                mov rdi, BUFFER_SIZE
135 00000106 0F05                      syscall
136
137
138                                  section .data
139 00000000 0000                    fd dw 0
140 00000002 3600000000000000013-    key: dq 0x36,0x13,0x92,0xa5,0x5a,0x27,0xf3,0x00,0x32
141 0000000B 000000000000009200-
142 00000014 000000000000A50000-
143 0000001D 00000000005A000000-
144 00000026 000000002700000000-
145 0000002F 000000F30000000000-
146 00000038 0000000000000000000-
147 00000041 003200000000000000
148 0000004A 0A                      endl: db 10
149 0000004B 0000000000000000        small_buffer: dq 0
150
151
152                                  section .bss
153 00000000 <res 000000B4>          file_buffer resb BUFFER_SIZE
```

```
CP1XSAT41X10^@^@^P^N^@^@^@^A^@^@^@^A^@^@^B^C^@^C<C2><F5>^@^Bf.^@^@pM^@^@^@
^@^@^@pGOBJECT_TRACKING: WARNING TRACKED OBJECT CROSSES MAIN TERRA COORDINATES^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@3&<80>
(END)
```