```
; taken from here cause I was banging my head against a wall for hours -> https://gist.github.
com/BertrandBordage/10921263
; checking if file exists -> https://gist.github.com/Archenoth/5380671
; This program will decrypt a packet from an incoming satallite
; It decrypts the packet in memory so there is no need to write subroutines to reverse the ord
er of the bytes
; Written by Jared Dyreson
; CPSC-240 TR @ 11:30 to 13:20

%define SYS_EXIT 60
%define SYS_READ 0
%define SYS_WRITE 1
%define SYS_OPEN 2
%define SYS_CLOSE 3
%define STDOUT 1
%define SYS_CREATE 85

%define BUFFER_SIZE 180

section .text
global  _start
_start:
  ; So we can read in our argument from argv[]
  add rsp, byte 0x10
  pop rdi
  jmp _check
_check:
; basic if/else control flow -> https://stackoverflow.com/questions/14292903/complex-if-statem
ent-in-assembly
  mov rdx,0
  cmp rdx,rax
  jle _cont
  jnle _exit_failiure
  _cont:

  ; open the file
  mov rax, SYS_OPEN
  mov rsi, 0
  syscall
  mov [fd], rax
  jmp _read_write

_read_write:
  ; Read the file into the buffer
  mov rax, SYS_READ
  mov rdi, [fd]
  mov rsi, file_buffer
  mov rdx, BUFFER_SIZE
  syscall

  cmp rax, 0
  je close_file

  jp _read_write


_exit_failiure:
  ; exit with code 1
  mov rax, 60
  mov rdi, 1
  syscall

close_file:
  ; Close the file stream
  mov rax, SYS_CLOSE
```

```
  mov rdi, fd
  syscall

  xor r8, r8
  xor rax, rax

  jmp decryptor
_reset_key:
  xor rax, rax
  jmp decryptor
decryptor:
  ; goal
  ; xor each byte in the file buffer, given a certain offset to that position in the string in
 the file_buffer and the key
  ; reset the key once we reach 9th element

  ; r8 -> indexing the file_buffer
  ; rax -> indexing our key
  ; r11 -> contains our needed variable from the file_buffer
  ; r12 -> contains our needed variable from the key

  ; if(r8 >= 180), we need to leave
  cmp r8, 180
  jge exit

  ; if(r9 > 8), we need to reset it
  cmp rax, 8
  jg _reset_key

  ; load the character from the file_buffer we need into the variable [check]
  lea rbx, [file_buffer]
  mov r11, [rbx+(r8*1)] ; variable = buff[i]


  ; move current offset into the correct register
  lea rbx, [key]

  cmp rax, 0
  je other
  jne _begin
  other:
  mov r12, 0x36 ; 0th index cannot be accessed for some reason
  jmp cont
  _begin:
  mov r12, [rbx+(rax*8)]; xor_key_variable = key[index]
  jmp cont
  cont:
  xor r11, r12 ; buf[i] ^ key[index]
  mov [file_buffer+r8], r11 ; r11 = buf[i] ^ key[index]
  inc r8 ; r8++
  inc rax ; rax++
  jmp decryptor ;loop back

exit:
  print_buffer:
  mov rsi, file_buffer
  mov rax, SYS_WRITE
  mov rdx, BUFFER_SIZE
  mov rdi, 1
  syscall
  mov rax, SYS_WRITE
  mov rsi, endl
  mov rdi, 1
  mov rdx, 1
  syscall
```

```
  jmp leave_segment
  leave_segment:
  mov rax, 60
  mov rdi, BUFFER_SIZE
  syscall


section .data
fd dw 0
key: dq 0x36,0x13,0x92,0xa5,0x5a,0x27,0xf3,0x00,0x32
endl: db 10
small_buffer: dq 0


section .bss
file_buffer resb BUFFER_SIZE
```