EXPLORER

main.asm U | atoi.asm U | readline.asm U | writeline.asm U | itoa.asm U | build.sh U

finalproject > readline.asm

OPEN EDITORS
- main.asm finalproject U
- atoi.asm finalproject U
- readline.asm finalp... U
- writeline.asm finalp... U
- itoa.asm finalproject U
- build.sh finalproject U

HOMEWORK
- AToI
  - Screenshot ATOI - atoi... U
  - Screenshot ATOI - mai... U
  - Screenshot ATOI - mai... U
- Exchange
- Fibonacci
- FibonacciLoop
- FibRecursive
- finalproject
  - atoi.asm U
  - atoi.o U
  - build.sh U
  - itoa.asm U
  - itoa.o U
  - main U
  - main.asm U
  - main.o U
  - readline.asm U
  - readline.o U
  - writeline.asm U
  - writeline.o U
- PrintLine
- .gitignore M
- compile.sh
- FibonacciFunction
- FibonacciFunction.cpp
- main
- main.asm

OUTLINE

TIMELINE

```asm
global readline
extern _read

section .text
; readline(buffer, size)
; Reads a line of text from stdin into the buffer.
; Arguments:
;   RDI = pointer to buffer
;   RSI = maximum number of bytes to read, buffer size
; Returns:
;   Buffer is filled with null-terminated string
readline:               ; save registers that will be used (callee-saved)
    push rbx
    push rcx
    push rdx
    push r8
    push r9

    mov rdx, rsi        ; rdx = number of bytes to read
    mov rsi, rdi        ; rsi = buffer pointer
    mov rdi, 0          ; rdi = file descriptor 0 (stdin)
    call _read          ; call _read system call
    ; RAX = number of bytes read
    ; check for errors or zero bytes read
    cmp rax, 0          ; compare bytes read with 0
    jle .done           ; if less or equal to 0, done

    ; null-terminate the string
    mov rcx, rax        ; rcx = number of bytes read
    dec rcx             ; decrement to get last character index
    mov bl, byte [rsi + rcx] ; load last character
    cmp bl, 10          ; check if last character is newline
    jne .skip_newline   ; if not newline, skip
    mov byte [rsi + rcx], 0 ; null terminate
    jmp .done           ; jump to done

.skip_newline:
    mov byte [rsi + rax], 0   ; if no newline, null terminate after last char

.done:                  ; restore saved registers
    pop r9
    pop r8
    pop rdx
    pop rcx
    pop rbx
    ret                 ; return to caller with buffer filled null-terminated string
```