**Manual Compilation & Linking**

1. Writing the Assembly code in prog.asm
2. Manually compiling it with nasm:
3. nasm -f elf32 -g -F stabs -l lst.l -o obj.o prog.asm
4. Then manually linking it with ld:
5. ld -m elf\_i386 -o exe.x obj.o

**Why?**

This approach helps **beginners understand**:

* What NASM and LD do separately
* The role of object files (.o)
* The process of converting human-readable code into an executable

**Using a Makefile (Automated with Makefile)**

Now, instead of running those two commands every time, a Makefile is introduced:

|  |
| --- |
| exe.x: obj.o ld -o exe.x obj.o    obj.o: prog.asm nasm -f elf -g -F stabs -l lst.l -o obj.o prog.asm |

Then you run: make

**Why?**

* To **automate** the build process
* To **avoid repetition** of long commands
* To **ensure dependencies** are respected (e.g., only recompiling if the source changes) • It's standard in real-world software projects

Steps for automating

* 1. mkdir folder1
  2. cd folder1
  3. vim prog.asm
  4. type :-

|  |
| --- |
| section .data    section .bss  buf resb 1  section .text  global \_start    \_start:  mov eax, 3 ;sys call mov ebx, 0 mov ecx, buf mov edx, 1 int 0x80    mov al, Byte[buf] inc al  mov byte[buf],al    mov eax, 4 mov ebx, 1 mov ecx, buf mov edx, 1  int 0x80    mov eax, 1 mov ebx, 0 int 0x80 |

* 1. nano Makefile
  2. inside it :-

exe.x: obj.o ld -m elf\_i386 -o exe.x obj.o

# obj.o: prog.asm nasm -f elf32 -g -F stabs -l lst.l -o obj.o prog.asm

1. To Save the File:

Press Ctrl + O → then Enter to save

Press Ctrl + X to exit

1. make
2. ./exe.x
3. Type a letter
4. If we typed ‘a’ , then it will print ‘b’ .(the next letter)