Reverse Engineering(study and plan note)

analyse pe or elf or exe, o file to get flag

== binary file

machine language = human readable

assembly language = machine readable etc 0x55 == push ebp

assemble

assemble language -> machine language

compile

c language etc -> assemble language -> machine language

link(linker)…integrate libraries

1.dynamic link… link when activation

2.static link… link when producing binary

symbol…the name used for recognizing by linker

memory map … .text .data .bss .heap .shared object .stack

3 way to analyze

1. Surface Analysis

… no need to implement code, collect data from past data sets and check

2. Dynamic Program Analysis

…need to implement program using debugger. better for Linux or windows environment?  
tool…gdb(Linux), Ollydog(Windows)

3. Static Program Analysis

…need inverse assemble, need knowledge (instruction such as move…)  
For step,

Start Surface Analysis to get rough bug -> use dynamic Analysis for each step

-> if still cannot understand what is bug or flag -> use Static Program Analysis for the part because it is difficult and taking time to read every assembly code

Personally, I’m using Mac and interested in ios application so do “crackme” Reverse Engineering challenges instead of CTF ?