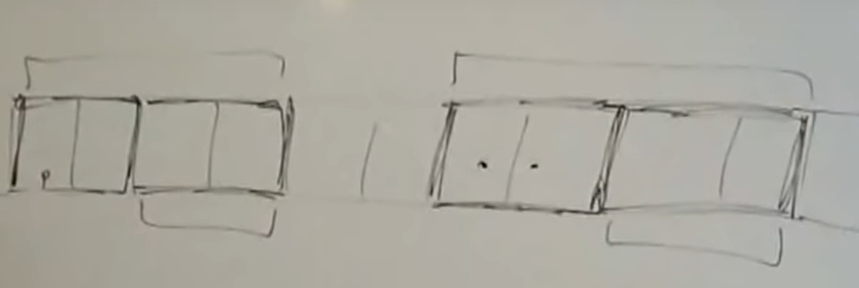
Sending data to keyboard isn’t that hard but receiving is hard.

Memory parts (dark squares) in order:

* Kernel (binary code – kernel.bin) – Code segment
* some data it uses – Data segment
* …
* program you are executing as user – Code segment
* data of your program – Data segment



Common approach of attacking a computer was to make a program load binary code into data of your program section and then jump into that and executing a virus or sth.

Nowadays OS don’t allow programs to jump into data segment.

Assume your CPU is executing your program and you pressed the keyboard. Also interrupt for keyboard press might be at kernel section.

CPU is limited to use only program and its data section part so it cannot use kernel part.

We have to set up an interrupt descriptor table IDT which contains the information for keyboard interrupt.

Global Descriptor Table: table of segments. Starting point of the segment and length of it. We also have flags for access rights, is it data or code segment, is it executable, …

1 entry is 8 bytes long.

* 2 bytes for length (limit)
* 3 bytes for starting point pointer
* 1 byte for access rights
* byte number 6 is divided into two half bytes
  + low 4 bits is for 4 more bits for the length
  + other 4 bits is for flags
* 1 byte for pointer

A drawing of a car

Description automatically generated with low confidence

pointer and limit are spread through the entry. This is bad.

Now we have 2 more files:

* gdt.h
* gdt.cpp

We have 20 bits for limit.

If we want to address the whole RAM, for example, then this 20 bits will not be sufficient.

limit is multiplied with 212.

So you have 20 bits + additional 12 virtual bits.

We get uint32\_t as parameter for the constructor. 2.5 most significant bytes (20 bits) will be put into limit at the entry. Rest (12 bits) wll be cut off but this is only legal if they are all 1s.

If 12 bits is not all 1, we decrease most significant 20 bits by 1. Then we can increase these 12 bits to all 1s. Worst can happen is unused, empty data segment.