1 OPERATING SYSTEMS LABORATORY XII - I/O Efficiency

Examples&Exercises:

- Compile and run the code.
- Analyze the code and output.

1. Memory layout of devices, all the devices in the linux are represented as file descriptors and these file descriptors have addresses in the memory;

    - Study the command

      ```
      $ cat /proc/ioports
      ```

        - How it looks like? Try to understand the devices and describe the order of appearance.
        - Each entry specifies (in hexadecimal) a range of ports locked by a driver or owned by a hardware device.

    - Study the command

      ```
      $ cat /proc/iomem
      ```

        - Similar to what happens for I/O ports, I/O memory information is available.
        - Once again, the values shown are hexadecimal ranges, and the string after the colon is the name of the ”owner” of the I/O region.

    - As far as driver (device driver) writing is concerned, the registry for I/O memory is accessed in the same way as for I/O ports, since they are actually based on the same internal mechanism.

2. Optimize file i/o performance; a program to show the effect of the buffer. code53.c (do not forget to download also the file, ourhdr.h)

    - Many applications assume that standard input is file descriptor 0 and standard output is file descriptor 1. In this code we use the two defined names STDIN_FILENO and STDOUT_FILENO from $< unistd.h >$.
    - The program does not close the input file or output file. Instead it uses the fact that whenever a process terminates, all open file descriptors are closed.

- This program works for both text file and binary files, since there is no difference between the two to the kernel.
- First create an input file by

  ```
  $dd if=/dev/zero of=inputfile bs=8K count=100
  ```

  - The output is

    ```
    100+0 records in
    100+0 records out
    819200 bytes (819 kB) copied, 0.010789 seconds, 75.9 MB/s
    ```

  - timing for the writing speed is given as the last item (as 75.9 MB/s),
  - what are the parameters *bs* and *count*? (man dd)
  - observe the writing speed by changing the value of *bs*,
  - observe the writing speed by changing the value of *count*,
- Now, execute the program as

  ```
  $ time code53 < inputfile > outputfile
  ```

  - Run the program using the different values for the BUFFSIZE. Fill the following table (inputfile size as 100 MB);

BUFFSIZE (byte)	User CPU (sec)	System CPU (sec)	Clock Time (sec)	# of loops
1				
2				
4				
8				
16				
32				
64				
128				
256				
512				
1024				
2048				
4096				
8192				
16384				
32768				
65536				
131072				