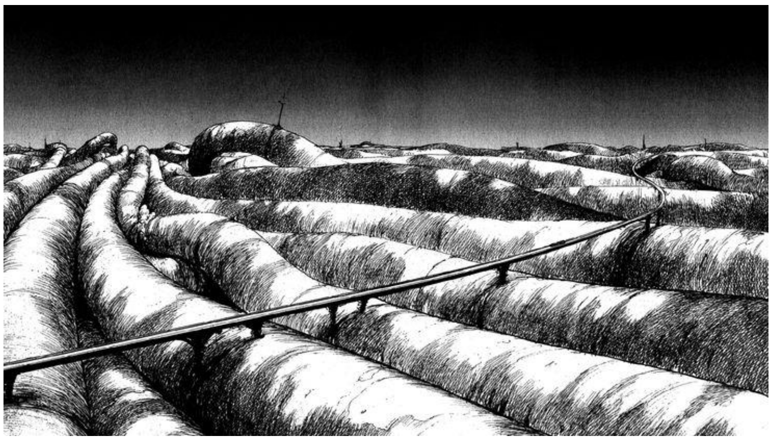
Lab Assignment
CPE 357
Systems Programming

CAL POLY
SAN LUIS OBISPO

Lab 7: DYI Pipe



Exercise:

Lets do our own pipe! (Not too big of an assignment due to prog 4)

For that, we need a FIFO structure. While lists can offer a solution, arrays might be the better idea here. If we have an array of a certain size, we need two (!) integer variables storing the information of where the occupied area of the pipe starts and where it ends.

- Think about it! FIFO! Start and end are on the same spot right at the beginning: 0.
- Then, someone writes bytes to it: end increases.
- Now, someone reads bytes from it: start increases (but cannot pass by end!)
- Respect carryovers when end or start reaches size!

Use following struct and write following functions

```
typedef struct mypipe
       {
       byte* pipebuffer;
       int buffersize;
       int start_occupied;
       int end_occupied;
       } mypipe;

//initializes (malloc) the pipe with a size of "size" and sets start and end.
void init_pipe(mypipe* pipe, int size);

//writes "size" bytes from buffer into the pipe, returns size
int mywrite(mypipe *pipe,byte* buffer, int size);

//reads "size" bytes from pipe into buffer, returns how much it read (max size), 0 if
pipe is empty
int myread(mypipe* pipe, byte* buffer, int size);
```

How we test

In the void main:

```
char text[100];
mypipe pipeA;

init(&pipeA, 32);
mywrite(&pipeA, "hello world", 12);
mywrite(&pipeA, "it's a nice day", 16);

myread(&pipeA, text, 12);
printf("%s\n", text);
myread(&pipeA, text, 16);
printf("%s\n", text);

mywrite(&pipeA, "and now we test the carryover", 30);
myread(&pipeA, text, 30);
printf("%s\n", text);
```

Bonus:

100 respect points for whoever knows the artist of the pipe-illustration above.