# Assignment 3 – Memory layout and pipe

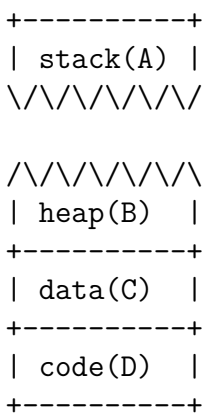
## Question 1: Process memory layout (4 points)

Here is a piece of program:

```
int str_len = 14;

int main() {
   char* str = malloc(str_len);
   memcpy(str, "hello world!\n", str_len);
   printf("%s", str);
   free(str);
}
```

When being executed, the program will be loaded into to memory. Below is a simplified memory layout of the running program (namely, a process).




**Please write down which part of the memory (i.e., A–D) do the following items locate.**
- variable "`str_len`":

- variable "`str`":

- memory pointed by "`str`" (or "`*str`"):

- the main function:

```
hint: to answer the questions, you need to understand
(1) what are local/global variables and dynamically allocated memory,
and (2) where they locate in memory.
```

# Question 2 Pipe (4 points)

Read the following code about syscall pipe and answer questions:

```c
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int
main(int argc, char *argv[])
{
    int pipefd[2];
    pid_t pid;
    char buf;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <string>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    if (pipe(pipefd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {    // child
        close(pipefd[1]);      // <---HERE

        while (read(pipefd[0], &buf, 1) > 0)
            write(STDOUT_FILENO, &buf, 1);

        write(STDOUT_FILENO, "\n", 1);
        close(pipefd[0]);
        exit(EXIT_SUCCESS);
```

```
    } else {            // parent
        close(pipefd[0]);          // Close unused read end
        write(pipefd[1], argv[1], strlen(argv[1]));
        close(pipefd[1]);          // Reader will see EOF
        wait(NULL);                // Wait for child
        exit(EXIT_SUCCESS);
    }
}
```

**Questions:**

2.a (1 point) What does this program do?
**Explain in 1–2 sentences.**

2.b (1 point) **If you comment out the line with "<---HERE", what has changed when you compile and rerun the program?**
`hint: notice if the program finishes.`

2.c (2 points) Why does commenting the line with "`<---HERE`" have this consequence?
**Explain why in 1–2 sentences.**