OS Assignment - 2

Team:

1- **Srinivas Piskala Ganesh Babu(spg349) – N13138339**
2- **Nanda Kishore Kalidindi (nkk263) – N16138926**

➔ Question 0: Getting Started with Xv6 OS

The Xv6 OS was built and run on os mentioned below using qemu

- Linux
- MAC

➔ Question 1: Hello World (Part 1)

- **Hello.c with only print**

Code:

```
#include "user.h"
#include "stat.h"
#include "types.h"
int main (void)
{
printf(1,"Hello World\n");
exit();
}
```

Output:

```
start 58
init: starting sh
$
$
$ hello
Hello World
$
```

- **Hello.c with MemoryAllocation**

Code:

```
#include "user.h"
#include "stat.h"
#include "types.h"
/*
 * Function: MAIN
 * - Stores the string HelloWorld! in the memory
 * - Outputs the data from the memory
 * - Frees the memory
 */
int main(void)
{
int i;
char *a;
a = (char *) malloc(11);
char *start;
start = a;
printf(1,"Start Address is : %d\n",start);
char b[11] = "HelloWorld!";
for(i = 0; b[i] != '\0'; i++)
   {
       *a = b[i];
       printf(1,"Next Address is : %d\n",a);
       a++;
   }

   *a  = '\0';
for(a = start; *a != '\0'; a++) {
  printf(1,"The Data written is : %c at %d\n",*a,a);
}
a = start;
printf(1,"The Data is : %s\n",a);
free(a);
printf(1,"The Data Fetch after Memory Freeing is : %s\n",*a);
printf(1,"The Address after Memory Freeing is : %d\n",a);
exit();
}
```

Output:

```
srinivas@srinivas-Lenovo-Flex-2-14:~/Desktop/xv6 OS/xv6-public$ sudo make qemu
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.0150637 s, 340 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 8.9412e-05 s, 5.7 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
354+1 records in
354+1 records out
181432 bytes (181 kB) copied, 0.000422566 s, 429 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive
file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2487
cat            2 3 14340
echo           2 4 13205
forktest       2 5 8067
grep           2 6 15892
init           2 7 14090
kill           2 8 13245
ln             2 9 13171
ls             2 10 16015
mkdir          2 11 13266
rm             2 12 13247
sh             2 13 24671
stressfs       2 14 14137
```

```
usertests      2 15 67093
wc             2 16 15026
zombie         2 17 12915
hello          2 18 12930
memoryalloc    2 19 14124
head           2 20 15238
stdread        2 21 14060
ten            2 22 15264
console        3 23 0
$ memoryalloc
Start Address is : 45040
Next Address is : 45040
Next Address is : 45041
Next Address is : 45042
Next Address is : 45043
Next Address is : 45044
Next Address is : 45045
Next Address is : 45046
Next Address is : 45047
Next Address is : 45048
Next Address is : 45049
Next Address is : 45050
Next Address is : 45051
Next Address is : 45052
The Data written is : H at 45040
The Data written is : e at 45041
The Data written is : l at 45042
The Data written is : l at 45043
The Data written is : o at 45044
The Data written is : W at 45045
The Data written is : o at 45046
The Data written is : r at 45047
The Data written is : l at 45048
The Data written is : d at 45049
The Data written is : ! at 45050
The Data written is : �at 45051
The Data written is : �at 45052
The Data is : HelloWorld!��
The Data Fetch after Memory Freeing is : D$ Hell�D$ oWor�D$ !�t&
The Address after Memory Freeing is : 45040
$ srinivas@srinivas-Lenovo-Flex-2-14:~/Desktop/xv6 OS/xv6-public$
```

➔ **Question 2 & 3: Implementing HEAD and Extending it (Part 2 and Part 3)**

- **Code:**

```
// Including Header Files
#include "user.h"
#include "stat.h"
#include "types.h"

// Buffer Declaration to store the data being read
char buf[2000];

/*
 * Function:  head
 * --------------------
 * Arguments:
```

```
 *     File Handle (Handle for Files and Zero for Pipeline)
 *     Number of Lines
 *     File Names
 *
 * Prints "n" lines of data read from a file :
 * n: number of lines that should be printed
 *
 * returns: pushes the selected number of lines to stdout
 *          prints error if n is 0
 */

void head(int handle, int lines, char *name) {
    int k,l,n;
    l=n=0;
    while((n = read(handle,buf,sizeof(buf))) > 0) {
      for(k=0;k<n;k++) {
        printf(1,"%c",buf[k]);
        if(buf[k]=='\n') {
             l++;
             if(l == lines) {
                 break;
             }
        }
      }
      if(l == lines) {
        break;
      }

    }
    if(n < 0){
      printf(1, "head: read error\n");
      exit();
    }
}

/*
 * Function: Number
 * --------------------
 * The Number Function: Converts the Number of lines argument to integer
 * (Handles the '-' sign present in the argument)
 *
 * Arguments:
 *   a[] : Character array of the argument string
 *
 * returns:
 *  Number of lines - integet type
 *  Default number of lines - 10 (In case of improper or no argument)
 *
 */

int number(char a[]) {
  int n;
  if(a[0] == '-') {
     n=atoi(a+1);
  } else {
     n = 10;
  }
return n;
}
```

```
/*
 * Function:  MAIN
 * --------------------
 * The MAIN Function: Inputs the data and call the HEAD Function
 *
 * Arguments:
 *  n: number of lines that should be printed
 *
 * returns:
 *  pushes the selected number of lines to stdout through HEAD Call
 *  prints error and usage help when there are no arguments
 *
 */

int main( int argc, char *argv[]) {

  int i,handle;
  int lines;

/*
// Argument: N number of lines missing - Error Condition

   if (argc <= 1) {
       printf(1,"Usage: head [LINES]... [FILE]..\n");
       printf(1,"Help: head 3 FILE --> Prints 3 lines from FILE\n");
       exit();
   }
*/

// Argument: Number of Lines Only

   if(argc <= 2){
     lines = number(argv[1]);
     head(0, lines, "");
     exit();
   }

// Argument: Number of Lines + File Names

    if (argc > 2) {
      lines = number(argv[1]);
      for(i = 2; i < argc; i++) {
        if ((handle = open(argv[i], 0)) < 0) {
          printf(1,"head: cannot open %s\n", argv[i]);
          exit();
        }
        head(handle, lines, argv[i]);
        close(handle);
      }
   }
  exit();
}
```

Output:

```
$
$ head -3 README
xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
but is implemented for a modern x86-based multiprocessor using ANSI C.
$
$
$
$ grep the README | head -3
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
xv6 borrows code from the following sources:
    JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)
$
$
$
$
$ cat README | head
xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer
to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,
2000)). See also http://pdos.csail.mit.edu/6.828/2016/xv6.html, which
provides pointers to on-line resources for v6.
$
$
```