COSC 350 System Software (Mini Test #2)

10/2/23

Name: ____________________________

1. (1.5 pt.)

Contiguous Allocation

- **Idea:** Files are stored in consecutive blocks.
- **Disadvantages:** Can lead to external fragmentation, and it’s difficult to resize files as they grow.

Linked-List Allocation

- **Idea:** Each file is stored as a linked list of blocks, where each block contains data and a pointer to the next one.
- **Disadvantages:** Random access is slow because the system must follow the pointers sequentially through the file blocks.

Linked-List Allocation with FAT

- **Idea:** The File Allocation Table (FAT) stores the pointers to the next blocks in a table rather than in each block.
- **Disadvantages:** The FAT must be loaded into memory, which can be slow for large disks, and still suffers from slow random access.

Index-Node Allocation (I-node)

- **Idea:** A structure (inode) contains pointers to the file’s blocks, allowing direct access to file data blocks.
- **Disadvantages:** Indirect blocks may be needed for large files, increasing the overhead of managing large files.

2. (0.5 pt.)
   Answer) System calls are run on kernel’s mode, it use kernal’s space but library function are run on user’s space, it use process’s own space.

3. (4 pt.)

```
#include <stdio.h>
#include <stdlib.h>
int st_to_int(char *); //function prototype
void main(int argc, char *argv[])
{
      int i, num;
      int esum =0;
      int osum =0;
      if (argc <= 1){// argment must be at least two or more
            printf("argument number error \n");
            exit(1);
      } //endif

      for (i=1; i<argc; i++){//read command line input
            num = st_to_int(argv[i]);
            if ((num % 2)== 0)
                  esum = esum + num;
            else
                  osum =osum+ num;
      }//end for
      printf("The sum of even arguments is %d\n", esum);
      printf("The sum of odd arguments is %d\n", osum);

  return;
} //endof program

/* convert numberical c-string to number */
int st_to_int(char *str)
{
  int num =0;
  int i =0;
  int negative = 0;
  if (str[0]== '-'){
        i =1;
        negative =1;
  }//end if
  if (str[0]== '+')
        i =1;

  while (str[i]!='\0')
  {
      num = 10 * num + (str[i] - '0');
      i++;
  }//end of while
  if (negative == 1)
     num = num * -1;
  return num;
}
```

4. (3 pt.)

```
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>

int main()
{
      int in, out, i;  //file descriptors of files
      char c; //currently read character
      off_t offset; //current offset
      int size; //file size

      in = open("foo", O_RDONLY); //open input file
      umask(0); //clear mask
      out = open("palindrome", O_WRONLY|O_CREAT, 00600);
      //rw-------

      while (read(in, &c, 1) ==1)
            write(out, &c, 1);
      //set offset to end of input file & get file size
      size = lseek(in, -1, SEEK_END)+1;
      for (i =1; i<=size; i++)
      {
            read(in, &c, 1);//read each char of input file
            write(out, &c, 1);  //and write to output file
            lseek(in, -2, SEEK_CUR); //offset to previous char
      }
      //close open files
      close(in);
      close(out);
      exit(0);
}
```

5. (1 pt.)

```
#include <stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int offset = 0;
    char a;
    int fd = open(argv[1],O_RDONLY);
    while(read(fd, &a, 1) == 1)
          offset++;
    printf("size of \"%s\" is %d bytes\n",argv[1], offset);
    close(fd);
    exit(0);
}
```