6/10 +1

COSC 350 System Software (Mini Test #3)

10/22/21

Name: Jung An

1. 3 pt.) Write a C program which accept an positive integer n as an argument and generates a structure of the length specified by n. The structure have three components, last name, first name id number. Fills number of information from the key board and display on stdout.

-2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main (int argc, char ** argv)
{
    int n= atoi(argv[1]);
    if (n < 0){
       printf("Argument error");
       exit(1);
    }

    struct s {                         ???
       char l-name[n];
       char f-name[n];                                        ??
       char id[n];
     }

     read(1,s.l_nane, n);
     read(1,s.f.nane, n);
     read(1, s.id , n);
     printf("last name = %s \n first name = %s \n ID=%s \n", s.l_nane, s.f_name, s.id);
```

2. (4 pt.) Write a complete c program as following.
    - A parent process creates a child process. The child create a child. Three processes are running concurrently.
    - The child process print “I am your child” 100 times and terminate.
    - The grandchild process run forever by printing “I am your grandchild”
    - Parent process run forever by printing “ I am your parent”.
    - Each process sleep one second after each print.
    - Parent process is waiting for the child’s termination. When the child is terminated, the parent print “I finish my job” and terminate itself.
    - The grandchild process terminate when the child process terminate. The grandchild process must find out it’s parent termination somehow.
    - Must not create zombie or orphan process.
    - Do not use signal!

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


int main( )
{ pid_t  pid,cpid;
   pid=fork();
    if(pid==0)
    {

      cpid=fork();
       if ((cpid==0)
        {
          pid_t  gcpid;
           gcpid=getppid();
            while (1)
            { if(gcpid !=1)
                ~~exit(0);~~
              else
                printf("I am your grandchild");
            }

        }
        else
        { int i=0;
          for(; i<100; i++)
              printf ("I am your child");
          exit(0);
        }

    }
    else
    { while (1)
        { printf("I  am your parent);
          if(wait(cpid)==1){
             Printf("I finished my job");
             exit(0);}
        }
   exit(0);
}
```

????

-2

if gcpid ==1 exit

you need use waitpid with option!

3. (3 pt.) Write a complete C program for sharing a file between parent and child. **This program receive an input file name as an argument** A input file will be opened only once by a parent before create a child by fork() system call. The parent open output file (named parent.txt with rw_rw_rw) and child open an output file (named child.txt with rw_rw_rw) for their output. Both parent and child processes try to create its own copy of input file. The parent and child process must synchronize to create a proper copy of input file.
   Do not use local variables only for child or only for parent since both has its own space each other.
   Do not use signal(), kill(), vfork() wait(), waitpid() or sleep(), pread() and pwrite() system call.

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


 int  main (int argc, char ** argv)
 {   int f_in, f_out;
    char buffer[1];
    int nread;

   f_in = open (f_in, O_RDONLY);
    int offset;
    int end = lseek(f-in, O, SEEK_END);
    pid-t pid;
    umask(0000);
    pid = fork();

    if (pid ==0)
     { f_out = Oopen ("child.txt", O_WRONLY || O_CREAT, 0666);
       offset = lseek(f-in, O, SEEK-SET);
       while (offset < end)
       {  lseek (f_in, offset, SEEK_SET);
          read (f_in, buffer, 1);
          write (f_out, buffer, 1);
          offset = lSeek (f-in, 0, curr);
       }
       close(f-out);
       exit(0);
    else
     { f_out = open("parent.txt", O_WRONLY || O_CREAT, 0666);
       offset = lseek (f-in, O, SEEK_SET);
       while (offset <end)
        {  lseek (f-in, offset, SEEK_SET);
           read (f-in, buffer, 1);
           write (f.out, buffer, 1);
           offset = lseek (f-in, O, curr);
        }
        close (f_out);
        exit(0)
    }
 close(f-in);
 exit (0);
}
```

7.