SIMLAB 2

Instruccions:

- Makefile:
  - Heu de crear un arxiu Makefile per compilar tots els programes de l'examen, usant regles independents per cadascun.
  - Inclou una regla clean per eliminar tots els binaris, arxius objecte i/o fitxers temporals.
  - Els programes han de compilar-se si i només si s'han actualitzat els seus arxius de codi font.
- Control d'errors i funció Usage. Tots els programes:
  - Han d'incloure un control adequat dels arguments usant la funció Usage().
  - Han de controlar els errors en TOTES les crides al sistema (excepció: write per pantalla)

Exercici 1: number_filter.c

Creeu un programa en C amb les següents característiques:

- Rep un sol paràmetre d'entrada *num_numbers* (enter) per línia de comandes
- Crea una unnamed pipe
- Crea un procés fill amb accés a la pipe
- El procés pare:
  - Genera num_numbers aleatoris (podeu utilitzar la funció rand())
  - Envia aquests números per la pipe
  - Espera al final del procés fill
- El procés fill:
  - Llegeix els números que li envia el pare per la pipe
  - Determina si el número rebut és parell o senar
  - Escriu els números en dos fitxers de sortida, `even_numbers.txt` i `odd_numbers.txt`, els números parells en un fitxer i els senars en un altre.
  - Els números s'han d'escriure en format string / ASCII (no binari) i un a cada línia del fitxer.

- Els fitxers de sortida s'han de crear si no existeixen, i han de tenir els permisos necessaris per a poder-hi llegir i escriure
- Compta quants números ha rebut, i mostra un *warning* per pantalla si no ha rebut el número esperat.

Solució:

- *Primer creem una unnamed pipe*
- *Després creem el procés fill amb fork*
- *El procés pare envia per la pipe sempre enters*
- *El procés fill converteix int a ASCII abans d'escriure als fitxers de sortida*
- *NOTA: el codi adjunt correspon a un sol fitxer number_filter.c, s'ha separat per claredat*

Codi comú pare i fill

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

    int pipe_fd[2];
    int num_numbers, random_no, ret, outval;
    int count_rx = 0;
    char buffer [128];
    int out_odd, out_even, i;



    //usage
    if (argc!=2) usage();

    num_numbers = atoi(argv[1]);


    if( pipe(pipe_fd)< 0) err_exit("Error open unnamed pipe");

    ret = fork();

    if (ret < 0){
        err_exit("Error fork");

    }else if (ret == 0){
        //Fill
```

Codi procés fill

```
}else if (ret == 0){
    //Fill

    close(pipe_fd[1]);
    //obrir fitxers sortida

    out_even = open("even_numbers.txt", O_WRONLY |O_CREAT, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
    out_odd = open("odd_numbers.txt", O_WRONLY |O_CREAT,   S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
    if (out_odd < 0 || out_even < 0 )  err_exit("Error open output files");

    //llegir pipe
    ret = read(pipe_fd[0],&outval,sizeof(outval));
    while (ret > 0){
        count_rx++;

        sprintf(buffer, "%d\n", outval);

        if (( outval % 2) == 0 ){
            //Format as strings
            write(out_even, &buffer, strlen(buffer));

        }else{
            write(out_odd, &buffer, strlen(buffer));

        }

        ret = read(pipe_fd[0],&outval,sizeof(outval));

    }

    if (count_rx != num_numbers){
        sprintf(buffer, "Warning: received unexpected total of values. Expected %d, Received %d\n", num_numbers, count_rx);
        write(1, buffer, strlen(buffer));
    }


    close(pipe_fd[0]);
    close(out_odd);
    close(out_even);
```

Codi procés pare

```
}else{
    //Pare
    close(pipe_fd[0]);
    sprintf(buffer, "Procés pare amb PID: %d, PID fill %d\n", getpid(), ret);
    write(1, buffer, strlen(buffer));

    for(i=0; i< num_numbers; i++){
        random_no = rand();
        sprintf(buffer, "Pare: sending number %d\n", random_no);
        write(1, buffer, strlen(buffer));
        write(pipe_fd[1],&random_no, sizeof(random_no));
    }



    close(pipe_fd[1]);

    //Requiered otherwise son may exit before
    waitpid(-1, NULL, 0);

}
```

Exercici 2: mini_pipe.c

Escriviu un petit programa que connecti amb una pipe dos processos que ens donen com a paràmetres d'entrada. El funcionament és el mateix que l'operador | de la terminal. Més en detall:

- Per línia de comandes ens donen: `mini_pipe proc1 param1 proc2 param2`
- param1 és un paràmetre del proc1, i param2 de proc2
- El programa mini_pipe crea una pipe i fa un fork
- El procés pare:
  - Connecta stdout a pipe
  - Muta a proc1
- El procés fill:
  - Connecta pipe a stdin
  - Muta a proc2
- El programa NO ha de funcionar amb un número arbitrari de paràmetres de proc1 i proc2, només amb un paràmetre per procés
- Exemple del funcionament:

```
$mini_pipe ls -l grep mini
```

```
-rwxrwxr-x 1 user user 16544 de des.   5 19:41 mini_pipe
-rw-rw-r-- 1 user user  1623 de des.   5 19:41 mini_pipe.c
```

Solució:

1. *Primer es crea una unamed pipe per connectar els dos processos*
2. *Després es crea un procés fill.*
3. *Triem que el procés pare muta a proc1 i el procés fill a proc2*
4. *Els dos processos, abans de mutar, redireccionen les pipes de manera que:*
   1. *stdout procés pare escriu a la pipe*
   2. *stdin procés fill llegeix la pipe*

```c
    if( pipe(pipe_fd)< 0) err_exit("Error open unnamed pipe");

    ret = fork();

    if (ret < 0){
        err_exit("Error fork");

    }else if (ret == 0){
        //Fill
        //tancar extrem escriptura
        close(pipe_fd[1]);
        //redirecciona lectura pipe a stdin
        dup2(pipe_fd[0], 0);
        close(pipe_fd[0]);
        //mutar
        execlp(argv[3], argv[3], argv[4] , (char *) 0);
        sprintf(buffer,"error mutar fill a %s", argv[3]);
        err_exit(buffer);

    }else{
        //Pare

        sprintf(buffer, "Procés pare amb PID: %d, PID fill %d\n", getpid(), ret);
        write(1, buffer, strlen(buffer));


        //tancar extrem lectura
        close(pipe_fd[0]);
        //moure write de pipe a stdout
        dup2(pipe_fd[1],1);
        close(pipe_fd[1]);

        execlp(argv[1], argv[1], argv[2] , (char *) 0);
        sprintf(buffer, "error mutar pare a %s", argv[1]);
        err_exit(buffer);
        close(pipe_fd[1]);



    }
```

Entrega

Entregueu un únic fitxer tar.gz al racó, amb el codi font i el Makefile. Per generar-lo podeu usar la comanda següent:

```
tar zcvf simlab2.tar.gz number_filter.c mini_pipe.c Makefile
```