Good info para esto:

fork: devuelve rc==0 si es hijo

execvp: reemplaza el proceso actual por el nuevo

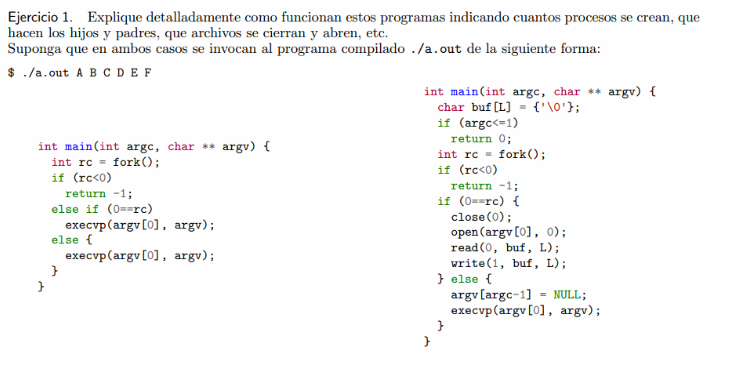
file descriptor: entero para identificar archivos abiertos. 0 para in, 1 para out, 2 para error

close(fd) cierra el fd asociado

open (args,flag): abre los archivos pasados en args con flag de lectura=0,escritura=1 o LyE=2

read(fd,buffer,cantdebytes): lee del fd asociado y guarda en buffer

write(fd,buffer,cantbytes): escribe en fd lo que hay en buffer



programa 1 programa 2

| nunca se terminaria de ejecutar ya que nunca se consumen los argumentos.  lo que ocurriria es que siempre que se logre ejecutar el fork, el proceso hijo reemplaza con execvp al proceso actual y todo volveria a empezar. el proc padre haria lo mismo. habria 1 hijo x proc. no se abre ni se cierra ningun archivo  fork bomb! | en la primera iteracion: el programa hijo (if 0==rc) cierra stdin, abre ./a.out solo con permisos de lectura. Lee lo que esta asociado al fd 0, lo agrega al buffer. luego escribe en stdout lo que estaba en el buffer. El programa padre se encarga de ir eliminando argumentos de ./a.out A B C D E F. ejecuta excecvp con los argumentos restantes reemplazando el proceso actual. esto ocurrira 6 veces (|a b c d e f| = 6).en la ultima iteracion termina con un return 0. |
| --- | --- |