# ESERCITAZIONE DI VENERDÌ 7/05/2021

1. Scrivere un programma **provaPipe.c** che, partendo dal programma provaopen.c visto a lezione (pag. 4 del file http://www.didattica.agentgroup.unimo.it/didattica/SOeLab/SessioniInterattive/PROVE_PRIMITIVE_AA2020-21.pdf), dopo la chiusura di fd1, invece che aprire di nuovo il file il cui nome viene passato come parametro, va a creare una pipe e quindi poi stampa i valori dei due file descriptor della pipe.
2. Scrivere un programma C **pipe-newSenzaClose.c** che, partendo dal programma pipe-new.c (http://www.didattica.agentgroup.unimo.it/didattica/SOeLab/SessioniInterattive/PROVE_PRIMITIVE_AA2020-21.pdf a pag. 76), non faccia nel figlio la chiusura del lato di scrittura della pipe e nel padre non faccia la chiusura del lato di lettura della pipe. **Verificare quindi che la mancanza delle chiusure dei lati delle pipe non utilizzati porta ad un comportamento scorretto!**
3. Scrivere un programma C **pipe-newGenerico1.c** che, partendo sempre dal programma pipe-new.c, assuma però un diverso protocollo di comunicazione che consideri la possibilità che il figlio invii al padre linee/stringhe di lunghezza generica: in particolare, il figlio, dopo aver letto una linea dal file passato come parametro (supposta non più lunga di 512 caratteri compreso il terminatore di linea), deve trasformare tale linea in stringa (come prima) e quindi deve mandare per prima cosa la lunghezza della stringa al padre e poi deve mandare la stringa al padre; quindi il padre, per ogni linea letta dal figlio, riceve per prima cosa la lunghezza della stringa e poi deve usare questa informazione per leggere il numero di caratteri che costituiscono la stringa, per poi stamparla analogamente all'esercizio originale.
   OSSERVAZIONE: per l'invio di queste due informazioni è opportuno usare la stessa pipe e NON crearne un'altra!
4. Scrivere un programma C **pipe-newGenerico2.c** che, partendo sempre dal programma pipe-new.c, assuma però un diverso protocollo di comunicazione che consideri la possibilità che il figlio invii al padre linee/stringhe di lunghezza generica: in particolare, il figlio, dopo aver letto una linea dal file passato come parametro (supposta non più lunga di 512 caratteri compreso il terminatore di linea), deve trasformare tale linea in stringa e quindi deve mandare la stringa al padre (come nell'esercizio originale); quindi il padre, per ogni linea letta dal figlio, deve ricevere via via i caratteri inviati dal figlio fino al terminatore di stringa e poi (solo dopo aver ricevuto TUTTA la stringa) deve stampare la stringa analogamente all'esercizio originale.
5. Scrivere un programma C **partec190201.c** che accetta un numero variabile di parametri che rappresentano nomi assoluti di file F1, F2 .. FN. Il processo padre deve generare N processi figli, ognuno dei quali associato ad un file. I processi figli eseguono concorrentemente leggendo dal file associato e comunicando al padre una selezione dei caratteri del file associato: i processi associati ai file dispari (F1, F3, ...) devono selezionare solo i caratteri alfabetici, mentre quelli associati ai file pari (F2, F4, ...) solo i caratteri numerici. Il padre deve scrivere i caratteri ricevuti dai figli su standard output, alternando un carattere alfabetico e uno numerico. Una volta ricevuti tutti i caratteri, il padre deve stampare su standard output il numero totale di caratteri scritti. Al termine, il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.
6. Scrivere un programma C **partec191201-a.c** che accetta un numero variabile di parametri (almeno 2) che rappresentano un nome di file F e singoli caratteri C1 ...CN. Il processo padre deve creare un numero di figli pari al numero di caratteri passati come parametri (N). Ogni processo figlio esegue in modo concorrente ed esamina il file F contando le occorrenze del carattere assegnato Ci: terminato il file, ogni figlio comunica al padre quante occorrenze di Ci sono presenti e termina. Il padre deve riportare sullo standard output il numero totale delle occorrenze specificando a quale carattere Ci si riferisce il conteggio. Al termine, il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.
   **NOTA BENE:** si usino N pipe (una per ogni processo figlio su cui viene scritto il numero di occorrenze del carattere Ci assegnato): in tale modo, il padre può individuare a quale carattere si riferisce il numero inviato!
7. Scrivere un programma C **partec191201-b.c** che accetta un numero variabile di parametri (almeno 2) che rappresentano un nome di file F e singoli caratteri C1 ...CN. Il processo padre deve creare un numero di figli pari al numero di caratteri passati come parametri (N). Ogni processo figlio esegue in modo concorrente ed esamina il file F contando le occorrenze del carattere assegnato Ci: terminato il file, ogni figlio comunica al padre quante occorrenze di Ci sono presenti e termina. Il padre deve riportare sullo standard output il numero totale delle occorrenze specificando a quale carattere Ci si riferisce il conteggio. Al termine, il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.
   **NOTA BENE:** si usi una sola pipe su cui ogni figlio scrive una struct con due campi (carattere e numero occorrenze); questa soluzione può essere accettabile visto che il testo non richiede che le informazioni inviate dal figlio vengano recuperate dal padre in un ordine specifico.
   **OSSERVAZIONE:** ***nei testi recenti viene specificato se va usata una struct o altra struttura dati!***