Università di Ferrara
Laurea Triennale in Informatica
A.A. 2021-2022
Sistemi Operativi e Laboratorio

Lab-04. Programmazione di Sistema

Prof. Carlo Giannelli

http://www.unife.it/scienze/informatica/insegnamenti/sistemi-operativi-laboratorio
http://docente.unife.it/carlo.giannelli
https://ds.unife.it/people/carlo.giannelli

Compilare Un File C

I file sorgenti scritti nel linguaggio C devono essere compilati per poter generare un file eseguibile.

Il compilatori C disponibile in laboratorio è **gcc**

GCC the GNU Compiler Collection

GCC (https://gcc.gnu.org/) è sicuramente uno dei compilatori più utilizzati nel mondo UNIX e Linux.

Permette di compilare sorgenti scritti nel linguaggio C ma anche altri linguaggi (Objective-C, D, Go, etc.)

Per compilare il file pippo.c usando gcc si usa il seguente comando da terminale:

gcc [-Wall -Wextra] -o pippo pippo.c

- **-Wall** abilita (quasi) tutti i messaggi di warning
- **-Wextra** abilita warning aggiuntivi
- **-o** specifica il nome del file eseguibile

Consigli per la compilazione

- **Attenzione**: omettendo l'opzione "-o <nomefile>", gcc produrrà un eseguibile chiamato “a.out”
- **Attenzione**: non passare il nome del file sorgente al parametro "-o", poiché quest’ultimo **verrebbe sovrascritto**
- Si consiglia di usare sempre le opzioni **-Wall -Wextra** (per gcc), per ottenere il maggior numero di informazioni possibile dal compilatore.

System Call UNIX

- Le principali system call viste finora sono:
  - operazioni sui file
    - **open()**
    - **read()**
    - **write()**
    - **close()**
  - **fork()** permette di creare un nuovo processo figlio
  - **wait()** aspetta la terminazione del processo figlio
  - **exit()** permette di terminare un processo
  - **exec...()**

Scrivere un file .c

- È importante non dimenticare di includere gli **header file**
- Per sapere quali header richiede una system call, si consulti la relativa man page

Ad esempio, nel caso di fork() (`man fork`), la man page restituisce:

```
SYNOPSIS
   #include <unistd.h>
   pid_t fork(void);
```

sarà quindi necessario includere il file header unistd.h

Esercizio 1 - Consegna

Esercizio molto semplice, ripasso di argomenti già visti in corsi precedenti.

Scrivere un programma che, utilizzando le opportune system call, permetta di **scrivere su un file delle stringhe fornite in input dall'utente**. Il programma deve presentare la seguente interfaccia di invocazione:

copy_input <nomefile>

dove nomefile è il nome di file sul quale scrivere i dati forniti in input.

Prima di tutto, il programma si deve occupare di creare il file <nomefile>. Se questo esiste, il programma deve sovrascrivere il suo contenuto.

Creato il file, il programma deve **mettersi in attesa di leggere una stringa dall'utente e scriverla su <nomefile>**. Il programma deve terminare quando l'utente inserisce la stringa "fine".

Esercizio 2 - Consegna

Scrivere un programma che **conti il numero di righe in cui le stringhe richieste dall'utente compaiono in un dato file**. Il programma deve presentare la seguente interfaccia di invocazione:

cerca <nomefile> <stringa1> ... <stringaN>

dove nomefile è un nome di file e stringa1...stringaN stringhe alfanumeriche.

Per ciascuna delle stringhe fornite dall'utente, il programma deve **creare un processo figlio** che si deve occupare di:

- **scrivere su un file** di log conteggi.txt (in append) una riga riportante l'operazione di conteggio nel seguente formato:
  "*<nomefile> <stringa>*"
- **contare il numero di righe** in cui essa compare all'interno del file nomefile e stamparlo a video.

Per la realizzazione del conteggio delle occorrenze delle stringhe all'interno del file *nomefile*, si faccia uso delle utility a riga di comando messe a disposizione dal sistema operativo.

Esercizio 2 - Traccia

cerca <nomefile> <stringa1> <stringa2>

- #include degli header file
- controllo degli argomenti e generazione dei figli necessari
- creo (e sovrascrivo se esiste) il file conteggio.txt
- chiudo il file conteggio.txt
- per ogni stringa, genero un processo figlio
  - se sono nel codice di un figlio
    - apro il file di log in append
    - riporto il log dell'operazione "<nomefile> <stringa>"
    - chiudo il file
    - eseguo il conteggio
  - se sono nel codice del padre attendo terminazione del figlio

Suggerimento: controllate utilizzando man i flag della system call open

Esercizio 3 - Consegna

Scrivere un programma interattivo che permetta all'utente di **visualizzare una "anteprima" dei file di testo nella directory corrente**. Il programma deve presentare la seguente interfaccia di invocazione:

anteprima <num>

dove **<num>** è un numero intero.

Per prima cosa, il programma si deve interfacciare con l'utente, leggendo da tastiera il nome del file di cui visualizzare l'anteprima. Dopo aver verificato che il file con il nome inserito dall'utente sia effettivamente esistente, **il programma deve quindi stampare a video le prime <num> righe del file richiesto**.

Al termine dell'operazione, il programma deve mettersi in attesa di una nuova richiesta da parte dell'utente e terminare se l'utente inserisce la stringa "fine".

Per la realizzazione della visualizzazione delle prime <num> righe dei file richiesti dall'utente, si faccia uso delle utility a riga di comando messe a disposizione dal sistema operativo.

Esercizio 3 - Traccia

anteprima <num>

- **#include** degli header file
- controllo degli argomenti
- finché l'utente non inserisce la stringa "fine:
  - controllo se il file specificato dall'utente esiste
    - genero un figlio:
      - (codice del figlio) - visualizzo il numero <num> di righe
      - (codice del padre) - attendo la terminazione del figlio