Debugging

- Se un programma compila non è detto che funzioni
- se un programma funziona non è detto che faccia quello che avete in mente;
- se un programma può essere eseguito, anche terminando prematuramente, si può usare un debugger;
- si possono monitorare le variabili e vedere il contenuto degli array;
- i più bravi possono guardare il contenuto dei registri;
- l'eseguibile deve contenere le informazioni sul sorgente;
- l'esecuzione deve poter procedere in modo da potersi fermare in corrispondenza ad ogni riga del sorgente;
- Per accedere al debugging si compila con

gcc -g -O0 program.c

Data Display Debugger ddd

- Scegliere *File* $\rightarrow$ *Open Program* e selezionare l'eseguibile;
- compare una finestra con il sorgente ed un'altra con i comandi;
- Run fa andare il programma dall'inizio alla fine;
- il programma si ferma ai breakpoint;
- Step va alla prossima linea del programma entrando in ogni funzione;
- Next fa lo stesso ma non entra nelle funzioni;
- Finish esce dalla funzione dove siete;
- Cont continua fino alla fine o fino al prossimo breakpoint;
- Until va alla linea del sorgente che sta più in basso di quella attuale (utile per uscire dai cicli)

Vedere le variabili

- Scegliere *Data* → *Display Local Variables*;
- puntare il mouse su di una variabile e tenercelo ne mostra il valore;
- facendo click col tasto destro del mouse su di una variabile si può vederne il valore in permanenza in una finestra;

Profiling

- Serve a misurare la velocità di esecuzione delle singole funzioni del programma;
- compilare con *gcc -pg sorgente.c*
- il programma deve essere poi esguito, quindi si da' il comando

  gprof eseguibile (| less)
- si ottiene una statistica del tempo speso in ogni funzione, e il numero di volte che ogni funzione è stata chiamata;
- il profiler non corregge i programmi, serve solo a renderli più veloci;
- guardare in quali funzioni un programma passa più tempo;
- concentrarsi su quelle chiamate più volte;
- si ottimizza un programma solo se già funziona, per poter verificare i risultati.

Shell di programmazione

- Permette di editare, compilare ed eseguire in uno stesso ambiente premendo un tasto;
- consente di gestire grandi progetti con molti file;
- accorcia il ciclo di sviluppo;
- ne esistono diverse sotto linux: tra queste kdevelop e anjuta;
- kdevelop è pensata per KDE e per programmatori C++;
- anjuta è pensata per GNOME e per il linguaggio C.

Iniziare un progetto

- Lanciate anjuta;
- scegliete *File* → *New project*;
- scegliete *generic / terminal* project;
- date un nome al progetto;
- scegliete il tipo di oggetto (eseguibile o libreria);
- scegliete il linguaggio di programmazione (C);
- descrizione del progetto;
- scegliete, se volete, il *copyright GNU* ma non *gettext*;
- *apply* crea ora il progetto;

Files del progetto

- Viene creato un file *main.c*;
- lo si edita con *File* → *Open* cercando nella cartella *src*;
- è un semplice programma *"Hello, World"*;
- compilate premendo *F9*;
- linkate premendo *F11*;
- eseguite con *F3*;
- provate ora a stampare π = *4.*atan(1.)*;
- scegliete *Settings* → *Compiler and linker settings* e *libraries* per includere le librerie matematiche;
- aggiungete un file di vostra scelta al progetto: ci potranno essere due *main*, uno dei quali va eliminato.