Corso di Fondamenti di Informatica 2
CdL Ingegneria Informatica
Ing. Franco Zambonelli

IL SISTEMA OPERATIVO UNIX: STRUMENTI PER LO SVILUPPO DI PROGRAMMI

Lucidi Realizzati in Collaborazione con:

Prof. Letizia Leonardi
Università di Modena

Prof. Antonio Corradi
Università di Bologna

Prof. Cesare Stefanelli
Università di Ferrara

Comandi per lo sviluppo di un programma

CC o GCC: Compilatore C (+ linker)

cc -o prog -g nomefile1 ... nomefileN

- I file oggetto hanno estensione **.o**
- Opzione **-o** → specifica nome dell'eseguibile (qui, **prog**) [default: **a.out**]
- I vari *nomefile1* ... *nomefileN* possono essere anche dei file oggetto (.o) → vengono solo linkati (non compilati)
- Il qualificatore **-g** aggiunge le tabelle per il debugger

---

cb: C program Beautifier
migliora indentazione e paragrafazione

---

cflow: crea il grafo di chiamate dei vari file in ingresso
cflow nomfilesorgente1 ... nomefilesorgente.n

- per default, solo funzioni
- con l'opzione -ix anche variabili extern e static

---

cxref: costruisce una tabella dei riferimenti incrociati
cxref nomefilesorgente

- dice chi viene chiamato da chi (in che file, a che riga, ...)

---

ctrace: controlla l'esecuzione
(attenzione: modifica il file sorgente)

Uso suggerito:

ctrace sorgente.c > temp.c {un file per volta}
cc temp.c { uscita di default su a.out }
a.out { esecuzione controllata }

---

dbx: Debugger Simbolico

I file eseguibili devono essere stati prodotti con l'opzione **-g** dal compilatore cc o gcc.

Comandi per la modifica:

list lineaIniziale, lineaFinale
edit nomeFile
edit nomeFunzione
func nomeFunzione {dichiara la funzione corrente}

Si possono analizzare e variare i dati:

display espressioneconvariabili *(watch del TurboC)*
undisplay espressioneconvariabili
whatis identificatore *{ dà il tipo della variabile }*
which identificatore *{ dà indicazioni sulla variabile correntemente specificata }*
whereis identificatore *{ fornisce indicazioni relative a variabili o entità con quel nome}*
dump funzione *{ stampa i valori ed i parametri della funzione stessa, se attiva }*

Esecuzione controllata:

run argomenti { inizia l' esecuzione }
stop at linea
stop if condizione { ad es. sul valore di una variabile }
stop in proc { varie forme di breakpoint }
cont { continua fino al prossimo breakpoint }
step n { *trace into*, esegue *n* linee per volta (def.1) }
next n { *step over*, idem come sopra}

Ad esempio, per partire:

stop in main
run

MAKE, il correlatore di strumenti

Il **make** gestisce in modo 'automatico' il progetto di file che costituiscono un'applicazione

Lo sviluppo diventa **automatico** dopo aver specificato le *dipendenze fra i file*