Configurarea mediului de lucru (IDE)

Aplicații precum DevCPP, Code::Blocks dar și alte medii integrate de dezvoltare bazate pe tehnologii *open-source*, pot fi configurate de către utilizator pentru a avea un comportament adecvat momentului în care este utilizat.

Uneltele de lucru open-source sunt cele bazate pe implementările GNU binutils, scrise special pentru mediile de lucru bazate pe SO Linux. Acestea susțin – și laolaltă cu GNU Compiler Collection (GCC) – formează o colecție de unelte esențiale proiectării software.
În ce ne privește, din suita GCC se remarcă compilatoarele gcc și g++, dedicate limbajelor C, respectiv C++. Ca informație generală, suportul GCC este mai larg, adică: începând cu versiunea 13.1, GCC include front-end-uri pentru C (gcc), C++ (g++), Objective-C și Objective-C++, Fortran (gfortran), Ada (GNAT), Go (gccgo), D (gdc, începând cu 9.1) și Modula-2 (gm2, începând cu 13.1). Extensiile de limbaj cu prelucrare paralelă OpenMP și OpenACC au fost acceptate începând cu versiunea GCC 5.1.

Din suita binutils fac parte următoarele unelte de dezvoltare software, foarte utile în diferite etape ale ciclului SW:

as	assembler popularly known as GAS (GNU Assembler)
ld	linker
gprof	profiler
addr2line	convert address to file and line
ar	create, modify, and extract from archives
c++filt	de-mangling filter for C++ symbols
dlltool	creation of Windows dynamic-link libraries
gold	alternative linker for ELF files
nlmconv	object file conversion to a NetWare Loadable Module
nm	list symbols exported by object files
objcopy	copy object files, possibly making changes
objdump	dump information about object files
ranlib	generate indices for archives (for compatibility; same as: ar -s)
readelf	display contents of ELF files
size	list section sizes and total size of binary files
strings	list printable strings
strip	remove symbols from object files
windmc	generates Windows message resources
windres	compiler for Windows resource files

Pentru mediile de lucru care au la bază SO Windows există o portare (preluare) din zona Linux, sub denumirea MinGW (Minimal GNU for Windows) care oferă astfel celor care folosesc acest SO posibilitatea de a scrie cod într-un unitar. Bazându-ne pe portabilitatea acestor limbaje, dar și evitând utilizarea unui API specific fiecărui SO, dezvoltarea unitară a aplicațiilor poate fi un deziderat posibil.

Astfel, pentru perioada de început a programării într-unul dintre limbajele (standardizate ISO) C sau C++, este util un comportament care să genereze informații suplimentare, adecvate, în detrimentul unui comportament implicit, care poate să nu ajute în procesul de asimilare a limbajelor de programare menționate.

Opțiuni de compilare (‚compilation flags')

Compilatoarele folosite, fie că sunt *open-source* sau *closed-source*, pot fi configurate. Prin *configurare* se înțelege stabilirea comportamentul în funcționare al acestora.
Orie compilator respectă anumite standarde de limbaj, și pentru aceasta oferă programatorului (utilizatorului) posibilitatea de a alege un anumit mod de lucru, în funcție de situație.
Însă nu toate compilatoarele respectă 100% un standard de limbaj oarecare. Din acest motiv, există un comportament al compilatorului care nu este 100% identic cu cel impus de standard (uneori acest comportament este denumit generic ‚standard de compilator'). Pentru a forța totuși utilizarea anumitor caracteristici ale standardului, acolo unde este posibil atât cât este posibil, pentru a reuși acest lucru programatorul are la dispoziție opțiunile de compilare oferite de către IDE/compilator (prin studiul manualului de utilizare sau a Help-ului integrat). Dacă nu, atunci se așteaptă o nouă versiune software a compilatorului, care este posibil să cuprindă și alte astfel de opțiuni.

`-Wall`	Activează mesajele de avertisment. Acest lucru permite *toate* avertismentele despre construcții pe care unii utilizatori le consideră îndoielnice și ușor de evitat (sau modificate pentru a preveni avertismentul). În mod *implicit*, o mare parte dintre avertismente NU este activă. De aceea, recomandarea este ca (cel puțin) pentru perioada de asimilare a sintaxei limbajelor C/C++ această opțiune să fie activă.
`-Wextra`	Activează mesaje de avertisment *suplimentare*, care nu sunt activate de utilizarea `-Wall` (prin activarea, în mod transparent, a unor flag-uri de compilare de tip care nu sunt activate de `-Wall`) **Obs.:** 1. Unele opțiuni de compilare din mulțimea`-Wall` și `-Wextra` pot activa anumite opțiuni specifice (cum ar fi `-Wunused`), care la rândul lor pot activa alte opțiuni înrudite (cum ar fi `-Wunused-value`). Este vorba despre un mecanism de activare (marcare) înlănțuită a unor opțiuni *înrudite*; 2. Această opțiune era numită `-W`. Deși denumirea veche este încă acceptată, se recomandă utilizarea noii denumiri.
`-Werror`	Toate mesajele de avertisment devin mesaje de eroare. Prin urmare, *nu vor putea fi ignorate*. Este o opțiune recomandată doar în perioada de început a studiului limbajelor C/C++, pentru a observa acele mesaje și a vă obișnui cu soluțiile specifice de rezolvare a fiecăruia. De exemplu, `-Werror-implicit-function-declaration` (inclusă odată cu această opțiune) va forța o avertizare (sau eroare) ori de câte ori o funcție este utilizată înainte de a fi declarată. Notă: ulterior, după ce se consideră că limbajul a fost asimilat din punct de vedere sintactic, se poate renunța la această opțiune.
`-g`	Forțează generarea informațiilor necesare depanării. Fișierul executabil final (care va depinde de sistemul de operare) poate crește în dimensiune, pentru a cuprinde această informație. Utilitarul de depanare (`gdb`) nu poate fi folosit fără aceste informații suplimentare.
`-pg`	Generează cod suplimentar pentru a include informații de profilare, potrivite pentru programul de analiză `gprof`. Trebuie să utilizați această opțiune atunci când compilați fișierele sursă despre care doriți date suplimentare și trebuie să o utilizați și la operația de editare a legăturilor (‚linking').
`-OX`	Stabilește nivelul de optimizare dorit: cu cât cifra X este mai mare, cu atât optimizarea devine mai ‚agresivă'. Nivelurile mai înalte de optimizare pot necesita timp suplimentar de compilare, în speranța reducerii timpului de execuție. Astfel: • La –O (`-O1`), se efectuează optimizări de bază, cum ar fi ‚îmbinarea constantelor' (‚constant merging') și eliminarea codului mort. Operațiunea de 'constant merging' presupune generarea *unei singure valori* în memorie, în

	situația existenței mai multor valori identice, pentru a putea fi folosită în acolo unde este nevoie în program. Pentru funcțiile mari se va folosi memorie suplimentară; • La –O2, sunt adăugate optimizări suplimentare, cum ar fi eliminarea subexpresiilor comune și utilizarea strictă a pseudonimelor (alias); • La –O3, se efectuează și mai multe optimizări, cum ar fi expandarea funcțiilor *inline* (C/C++) și vectorizarea (doar pentru C++); • `-Os` optimizează pentru mărimea executabilului (`size`); • `-O0` înhibă optimizarea (dezactivează orice fel de optimizare).

`-std=c99`
 	
`-std=c++11`
	Stabilește standardul (dialectul) limbajului C care va fi folosit la compilare. Va include sintaxa din standardul specificat (de ex. `C99`), cum ar fi `bool` (și alte caracteristici ale dialectului respectiv)
	Similar pentru limbajul C++ (la utilizarea `g++`).

Configurarea comportamentului unui IDE este recomandabil să se facă înainte de începerea lucrului cu acestea.
Cel puțin în situația unui mediu de lucru cu interfață de utilizare (interfață grafică) în orice moment se va putea reveni foarte ușor, folosind meniul dedicat (de obicei `Tools` → `Compiler options`) și se vor putea adăuga/șterge opțiunile de compilare dorite (cf. tabelului prezentat, dar și în plus/minus).

Multă baftă și inspirație în scrierea propriilor programe!