Περιβάλλον Unix και μεταγλώττιση αρχείων C++

#3

Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

Πώς μπορώ να έχω πρόσβαση σε περιβάλλον Unix;

- Εγκατάσταση διανομής Linux (προτεινόμενη διανομή: μια πρόσφατη διανομή Ubuntu LTS).
- Εγκατάσταση διανομής Linux παράλληλα με διατήρηση εγκατάστασης Windows (dual boot).
- Εγκατάσταση WSL2 (Windows Subsystem version 2 for Linux) σε υπολογιστή με εγκατάσταση Windows (10 ή 11).
- Εγκατάσταση Linux Virtual Machine (VM) σε VirtualBox ή VMWare.
- Χρήση Docker για τη δημιουργία ενός Linux container.
- Χρήση Vagrant για δημιουργία ενός Linux VM.

Βασικές εντολές στο Linux

ls	Λίστα με όλα τα αρχεία και τους καταλόγους στον τρέχοντα κατάλογο
cd <dir>	Αλλαγή του τρέχοντος καταλόγου σε <dir>
cd ..	Αλλαγή του τρέχοντος καταλόγου ένα επίπεδο πάνω στην ιεραρχία καταλόγων
cd ~	Αλλαγή του τρέχοντος καταλόγου στο home κατάλογο
mkdir <dir>	Δημιουργία του υποκαταλόγου <dir> στον τρέχοντα κατάλογο
rm <filename>	Διαγραφή του αρχείου <filename>
rm -r <dir>	Διαγραφή του καταλόγου <dir> των περιεχομένων του και όλων των υποκαταλόγων του
cat <filename>	Έξοδος του κειμένου του αρχείου <filename> στην οθόνη
chmod +x <filename>	Αλλαγή των ιδιοτήτων του αρχείου <filename> έτσι ώστε να επιτρέπεται η εκτέλεσή του.
more <filename>	Παρόμοιο με το cat, επιτρέπει σκρολάρισμα της οθόνης
man <keyword>	Εμφάνιση της σελίδας του εγχειριδίου (manual) για διάφορες προγραμματιστικές έννοιες και έννοιες του unix

Εγκατάσταση gcc σε Ubuntu 22.04.2 LTS

$ **sudo apt update**

$ **sudo apt install build-essential**

$ **sudo apt-get install manpages-dev**

$ **g++ --version**

g++ (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

Copyright (C) 2021 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Docker

Χρήση Docker Container για ανάπτυξη κώδικα σε C++

- **Πλεονεκτήματα**:
  - Ομοιόμορφο περιβάλλον ανεξάρτητο από τον host OS
  - Εύκολη αναπαραγωγή του περιβάλλοντος
  - Προεγκατεστημένα εργαλεία (g++13, cmake, gdb)
  - Δεν επηρεάζει το host σύστημα
- **Απαιτήσεις**:
  - Εγκατάσταση Docker Desktop
  - Visual Studio Code με Remote-Containers extension

Δομή Project με Docker

Δομή αρχείων:

```
project_folder/
├── .devcontainer/
│   └── devcontainer.json    # Ρυθμίσεις VS Code
├── Dockerfile               # Ορισμός περιβάλλοντος
├── main.cpp                 # Πηγαίος κώδικας
└── Fraction.cpp/hpp
```

Δύο βασικά αρχεία:

- Dockerfile → Περιγράφει το τι εγκαθίσταται
- devcontainer.json → Περιγράφει το πως χρησιμοποιεί το container το VS Code

Docker Environment Setup

- **Build του Docker image**:
  - docker build -t cpp-dev-environment .

    https://github.com/georgetzan/cpp-docker-dev-environment
- **Εκτέλεση Container**:
  - docker run -it --rm -v $(pwd):/workspace cpp-dev-environment
- **Ή με VS Code Dev Containers**:
  - 1. Άνοιγμα φακέλου στο VS Code
  - 2. Command Palette → "Dev Containers: Reopen in Container"
  - 3. Αυτόματη χρήση του .devcontainer/devcontainer.json

Projects C++ με διαμερισμό κώδικα σε πολλά αρχεία

Αρχεία επικεφαλίδων, αρχεία πηγαίου κώδικα

Projects πολλαπλών αρχείων (1/2)

- Αν και είναι δυνατόν οποιοδήποτε πρόγραμμα να γραφεί σε ένα μόνο αρχείο, συχνά τα προγράμματα διαμερίζονται σε πολλά αρχεία:
  - **Αρχείο επικεφαλίδας (header)**: περιέχει τη δήλωση της κλάσης, συνήθως έχει ως επέκταση .hpp (π.χ. circle.hpp).
  - **Αρχείο υλοποίησης**: περιέχει τον ορισμό της κλάσης, δηλαδή της υλοποίησης των συναρτήσεων μελών της κλάσης, συνήθως έχει ως επέκταση .cpp (π.χ. circle.cpp).
  - **Αρχείο οδηγός**: περιέχει την main() και πιθανώς άλλες συναρτήσεις που χρησιμοποιούνται στο πρόγραμμα, συνήθως έχει ως επέκταση .cpp (π.χ. main.cpp).

Projects πολλαπλών αρχείων (2/2)

- Λόγοι για τους οποίους χρησιμοποιούνται πολλαπλά αρχεία:
  - Επιτρέπει να γίνονται αλλαγές στην υλοποίηση των κλάσεων χωρίς να επηρεάζεται η διεπαφή της κλάσης.
  - Επιτρέπει σε μια κλάση να χρησιμοποιείται από πολλά προγράμματα οδηγούς.
  - Αυξάνει τις ευκαιρίες για επαναχρησιμοποίηση κώδικα.
  - Ο κώδικας γίνεται περισσότερο τμηματικός (modular).
  - Στην πράξη, πολλές φορές, δεν είναι δυνατόν να γραφεί κώδικας για ένα μεγάλο έργο λογισμικού σε ένα μόνο αρχείο.

Μεταγλώττιση

- Μεταγλώττιση είναι η διαδικασία μετατροπής ενός αρχείου από τη γλώσσα C++ σε γλώσσα που μπορεί να «καταλάβει» ο υπολογιστής, δηλαδή τη γλώσσα μηχανής του συγκεκριμένου υπολογιστή.
- Η μεταγλώττιση εμπεριέχει δύο κύριες φάσεις:
  - **φάση μεταγλώττισης** που είναι υπεύθυνη για τη μετάφραση της γλώσσας.
  - **φάση σύνδεσης** που είναι υπεύθυνη για την επίλυση των αναφορών μεταξύ επιμέρους αρχείων.

Φάση μεταγλώττισης (compile phase)

- Πραγματοποιεί τις οδηγίες του προεπεξεργαστή (preprocessor directives) όπως είναι οι: #include, #define κ.λπ.
- Ελέγχει τη σύνταξη του προγράμματος.
- Ελέγχει ότι οι συναρτήσεις και οι μεταβλητές έχουν δηλωθεί πριν τη χρήση τους.
- Δεν ελέγχει ότι οι συναρτήσεις που χρησιμοποιούνται έχουν οριστεί.
- Δημιουργεί αναπαράσταση κώδικα μηχανής σε μια μορφή που ονομάζεται αντικείμενο αρχείο (object file). Πρόκειται για ένα αρχείο με όνομα που έχει ως επέκταση .o
- Το αντικείμενο αρχείο δεν είναι εκτελέσιμο και μπορεί να περιέχει χρήση συναρτήσεων οι οποίες δεν έχουν οριστεί (δηλαδή δεν υπάρχει υλοποίηση για αυτές).

Φάση σύνδεσης (link phase)

- Συνδέει τα αρχεία αντικείμενα σε ένα εκτελέσιμο πρόγραμμα.
- Η φάση σύνδεσης είναι η φάση κατά την οποία οι κλήσεις συναρτήσεων συνδέονται με τους ορισμούς (υλοποιήσεις) των συναρτήσεων, και ο μεταγλωττιστής ελέγχει ότι υπάρχει ένας και μόνο ένας ορισμός για κάθε συνάρτηση που καλείται.
- Όλες οι συναρτήσεις μέλη θα πρέπει να έχουν οριστεί κατά τη διάρκεια αυτής της φάσης.
- Θα πρέπει να υπάρχει μια main() συνάρτηση έτσι ώστε το εκτελέσιμο να γνωρίζει από που να ξεκινήσει.

Παράδειγμα μεταγλώττισης project με διαμέριση αρχείων

- **g++ -c Fraction.cpp**
  - πραγματοποιεί τη φάση μεταγλώττισης στο αρχείο Fraction.cpp και δημιουργεί το αντικείμενο αρχείο Fraction.o
- **g++ -c main.cpp**
  - πραγματοποιεί τη φάση μεταγλώττισης στο αρχείο main.cpp και δημιουργεί το αντικείμενο αρχείο main.o
- **g++ Fraction.o main.o -o main**
  - συνδέει τα αρχεία αντικείμενα, ο κώδικας στο main.o που καλεί μέλη συναρτήσεις της κλάσης Fraction, συνδέεται με τους ορισμούς (υλοποίησεις των μελών συνατήσεων) που έχουν μεταγλωττιστεί στο αρχείο Fraction.o
- Συντόμευση που πραγματοποιεί μεταγλώττιση και σύνδεση σε ένα βήμα:
  - **g++ main.cpp Fraction.cpp -o main**

https://github.com/chgogos/oop/blob/master/various/COP3330/lect3/Fraction.cpp
https://github.com/chgogos/oop/blob/master/various/COP3330/lect3/Fraction.hpp
https://github.com/chgogos/oop/blob/master/various/COP3330/lect3/main.cpp

Αναφορές

- http://www.cs.fsu.edu/~xyuan/cop3330/
- https://www.docker.com/
- https://learn.microsoft.com/en-us/windows/wsl/install
- https://www.vagrantup.com/
- https://www.virtualbox.org/
- https://www.vmware.com/
- https://linuxize.com/post/how-to-install-gcc-compiler-on-ubuntu-18-04/
- https://github.com/georgetzan/cpp-docker-dev-environment