

# ΜΥΥ-402 Αρχιτεκτονική Υπολογιστών

## Οδηγίες git, GitHub

---

Αρης Ευθυμίου

Πηγές:

Timothy Wood - Git & GitHub: <http://faculty.cs.gwu.edu/~timwood/wiki/doku.php/learn:git>

Git Immersion: <http://gitimmersion.com/index.html>

# Το σημερινό μάθημα

---

- Τί είναι το Git και το GitHub
- Αποθετήρια
  - δημιουργία
  - κλωνοποίηση
- Στιγμιότυπα
- Κατάλογος εργασίας και στιγμιότυπα
- Απομακρυσμένα αποθετήρια
- Χρήση 2 τοπικών αποθετηρίων
- Αναίρεση αλλαγών
- Προχωρημένη χρήση - παρακλάδια



# Τι είναι το git

---

- Το git είναι ένα σύστημα διαχείρισης εκδόσεων αρχείων
  - Στα Αγγλικά: version control system
  - Παρακολουθεί τις αλλαγές σε ένα σύνολο αρχείων κρατώντας στιγμιότυπά τους
  - Το σύνολο στιγμιοτύπων ονομάζεται **αποθετήριο (repository)**
    - Όλα τα αρχεία και υποκατάλογοι κάτω από ένα «πατρικό» κατάλογο
- Χρήσεις:
  - επιστροφή σε μια παλιότερη έκδοση αν έχει γίνει κάποιο λάθος
  - δοκιμή παραλλαγών (ονομάζονται παρακλάδια - **branches**) χωρίς να επηρεάζεται η τρέχουσα κατάσταση των αρχείων



# Τι είναι το GitHub

---

- Το GitHub είναι ένας εξυπηρετητής που φιλοξενεί αποθετήρια και μια εύχρηστη διεπαφή ιστού
  - υποστηρίζει όλες τις εντολές του git
  - και παρέχει μερικές επιπλέον δυνατότητες
- Η εγγραφή χρηστών και η φιλοξενία ανοιχτών (public) αποθετηρίων είναι δωρεάν
  - άλλες χρήσεις χρειάζονται πληρωμή (ιδιωτικά αποθετήρια)
- Είναι εξαιρετικά δημοφιλές
  - εργοδότες υποθέτουν ότι αιτούντες εργασία θα το γνωρίζουν
  - ένας τρόπος να (απο/επι)δείξει κανείς τις ικανότητές του
    - συμμετοχή σε κοινά project, δημοσίευση project μαθημάτων, ...
- Υπάρχουν και άλλες αντίστοιχες υπηρεσίες
  - π.χ. BitBucket



# GitHub organizations

---

- Εκτός από αποθετήρια μεμονωμένων ανθρώπων παρέχονται αποθετήρια για οργανισμούς
- Όπως ο Uoi-CSE-MYY402
- Κάτι σαν υπερ-κατάλογος αποθετηρίων
- με ομάδες ανθρώπων να μοιράζονται αποθετήρια
- Διάφορα επίπεδα προσβασης
  - διαχειριστής, μόνο ανάγνωση, εγγραφή, ...



# Γιατί τα χρησιμοποιούμε

---

- Εξαιρετικά χρήσιμα ως εργαλεία
  - οργάνωση εργασιών, συνεργασία σε ομαδικά project, ...
  - de facto προαπαιτούμενα για εύρεση εργασίας
  - χρήσιμα για παράδοση εργασιών/ασκήσεων
- Το μάθημα απαιτεί ελάχιστη χρήση τους
  - μόνο παραλαβή, παράδοση κώδικα εργασιών
  - ελάχιστος χρόνος εκμάθησης (για απλή χρήση)
- Σκοπός
  - γνωριμία, βασική χρήση για το μάθημα
  - πειραματισμός, βαθύτερη γνώση για προσωπική χρήση και εξέλιξη δεξιοτήτων



# Εφαρμογές git, ρυθμίσεις

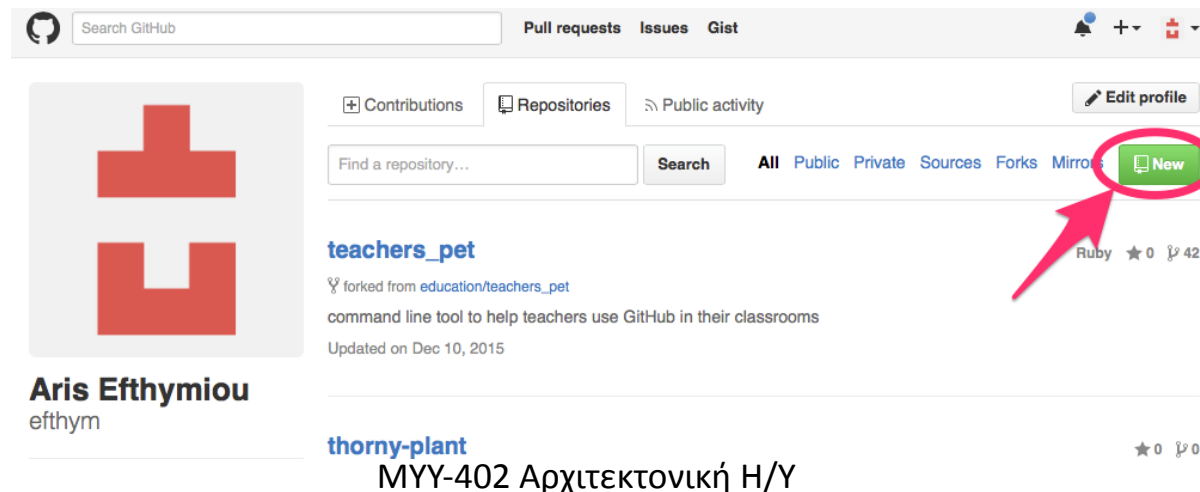
---

- Θα χρησιμοποιήσουμε το git από τερματικό
  - υπάρχουν και GUIs. Δείτε το φυλλάδιο του lab 0.
- Εγκατάσταση από <http://git-scm.com/downloads>
  - απλές οδηγίες, δε θα πρέπει να έχετε δυσκολίες
  - ήδη εγκατεστημένο στους υπολογιστές των εργαστηρίων
- Εγγραφή στο GitHub
  - <https://education.github.com/pack>
- Ρυθμίσεις
  - όνομα χρήστη και διεύθυνση email
  - `git config --global user.name "Your Name"`
  - `git config --global user.email "email@u"`
  - Για email χρησιμοποιείτε τη διεύθυνση που δώσατε στο GitHub



# Δημιουργία αποθετηρίου

- Για το μάθημα τα αποθετήρια θα είναι έτοιμα στο GitHub
  - θα χρειαστεί κλωνοποίηση για να έχετε «αντίγραφο εργασίας»
  - σε κάθε άσκηση θα παίρνετε ένα κατάλογο με τα επιπλέον αρχεία
- Στο GitHub, **στη σελίδα σας**, πατήστε το πράσινο κουμπί: “New (Repository)” και δώστε ένα όνομα
  - αυτό δημιουργεί αποθετήριο μόνο στο GitHub
  - δεν θα έχετε τοπικό αντίγραφο. Βλ. κλωνοποίηση
  - στο UoI-CSE-MYY402 δεν μπορείτε να δημιουργήσετε νέα αποθετήρια





# Δημιουργία αποθετηρίου

---

- Σε υπολογιστή (προσωπικό ή εργαστηρίου)
  - σε τερματικό, σε ένα κατάλογο:
  - `git init`
  - αυτό δημιουργεί μόνο τοπικό αποθετήριο, όχι στο GitHub
  - το αποθετήριο είναι άδειο ακόμη και αν υπάρχουν αρχεία στον κατάλογο



# Κλωνοποίηση αποθετηρίου

---

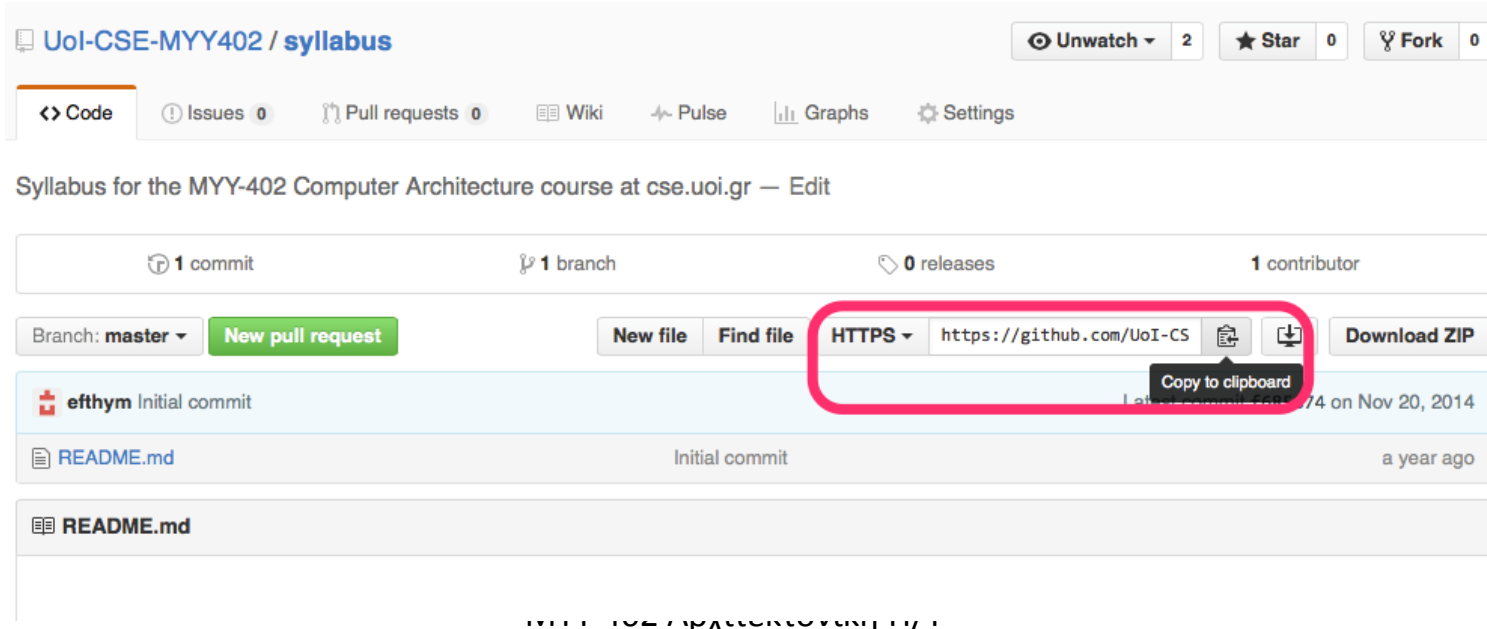
- Το αποθετήριο του GitHub είναι για αποθήκευση, συνεργασία
  - Δεν μπορείς να μεταγλωτίσεις, να τρέξεις προγράμματα, ...
- Χρειάζεται ένα τοπικό αντίγραφο, κλώνος (**clone**)
  - με αντίγραφο όλων των αρχείων του τρέχοντος στιγμιότυπου, στον υπολογιστή που έχετε μπροστά σας
    - είτε στο εργαστήριο ή στο σπίτι
    - ή και στα δύο (βλ. παρακάτω)



# Κλωνοποίηση αποθετηρίου

- Κλωνοποίηση:

1. βρείτε το URL του αποθετηρίου σας στο GitHub
    - μοιάζει με `https://github.com/USERNAME-or-ORGNAME/REPO.git`
  2. δημιουργήστε τον κλώνο:
    - Εντολή σε τερματικό: `git clone <URL>`
    - Θα δημιουργηθεί κατάλογος με το όνομα του αποθετηρίου
- Ο κατάλογος ονομάζεται κατάλογος εργασίας (**working-directory**)



Uoi-CSE-MYY402 / syllabus

Unwatch 2 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

Syllabus for the MYY-402 Computer Architecture course at cse.uoi.gr — Edit

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request

New file Find file HTTPS https://github.com/UoI-CS Copy to clipboard Download ZIP

efthym Initial commit

README.md Initial commit a year ago

README.md



# Στιγμιότυπα 1/2

---

- Το git κρατάει στιγμιότυπα μόνο όταν το ζητήσουμε
  - με την εντολή `git commit`
  - τα στιγμιότυπα ονομάζονται **commits**
- Τα commits καταγράφουν κάποιο «σημαντικό γεγονός»
  - την ολοκλήρωση κάποιου μέρους της δουλειάς
  - π.χ. αρχή του project, προσθήκη υπορουτίνας A, ...
  - το σημαντικό είναι υποκειμενικό: καλύτερα να υπάρχουν πολλά commits
- Απαιτείται ένα μικρό κείμενο που περιγράφει το γεγονός
  - ονομάζεται **commit message**



# Στιγμιότυπα 2/2

---

- Υπάρχουν πολλά commits και το git έχει τη δυνατότητα να τα βρεί
  - κάθε commit έχει ένα μοναδικό κωδικό, ονομάζεται **hash**
  - είναι ένας πολύ μακρύς δεκαεξαδικός αριθμός
  - πληροφορία στον κρυφό υποκατάλογο `.git` του working directory
- Υπάρχει πάντα ένα τρέχον στιγμιότυπο
  - που έχει το όνομα **HEAD**
  - το HEAD δείχνει σε άλλο στιγμιότυπο όταν κάνουμε commit
- Τα στιγμιότυπα σχετίζονται
  - εκτός από το αρχικό, κάθε στιγμιότυπο προκύπτει από ένα προηγούμενο με μερικές αλλαγές
  - σειρά αλλαγών μέχρι τώρα, εντολή: `git log`



# Κατάλογος εργασίας, στιγμιότυπα

---

Ποιά η διαφορά ενός «στιγμιότυπου» από το σύνολο αρχείων του καταλόγου εργασίας που βλέπω τώρα;

- Τα αρχεία που φαίνονται στον κατάλογο εργασίας...
  - μπορεί να είναι τα ίδια με ένα στιγμιότυπο (συνήθως το τρέχον),
  - αλλά μπορεί και να έχουν αλλάξει
- Επιπλέον μπορεί να υπάρχουν αρχεία **που δεν είναι** μέρος του στιγμιότυπου
  - Συνήθως παραγόμενα αρχεία: εκτελέσιμα, object, log κ.α.
  - ή αρχεία που μόλις προστέθηκαν και δεν έχουν κρατηθεί σε κάποιο στιγμιότυπο ακόμη
- Το αποθετήριο είναι, χονδρικά, το σύνολο των στιγμιοτύπων
  - πληροφορία κρατιέται στο .git του καταλόγου εργασίας
  - και στο GitHub (ή και αλλού)



# Προσθήκη σε στιγμιότυπο

---

- Γίνεται σε δύο στάδια
  - προσωρινή προσθήκη
    - εντολή: `git add`
  - τελική προσθήκη
    - εντολή: `git commit`
- Γιατί χωριστά στάδια:
  - ευελιξία για προχωρημένους χρήστες
- Χρήσιμη εντολή: `git status`
  - ποιά αρχεία «παρακολουθούνται» (έχουν γίνει `add`)
  - ποιά έχουν αλλάξει (είτε παρακολουθούνται, είτε όχι)
  - και πολλά άλλα



# Προσωρινή προσθήκη

---

- Ονομάζεται **staging** (ή add to Index)
- Είναι η στιγμή που κρατιέται το στιγμιότυπο
  - Αν ξανα-αλλάξει το αρχείο από τότε μέχρι την τελική προσθήκη, **οι αλλαγές δεν θα καταγραφούν** στο στιγμιότυπο.
    - αν δεν ξαναγίνει προσωρινή προσθήκη
- Απαραίτητη για κάθε αλλαγή σε αρχείο
  - όχι μόνο για νέα αρχεία που θέλετε να προσθέσετε στο commit
- Μπορεί να γίνει πολλές φορές πριν από τη τελική προσθήκη
- Εντολή: `g i t add <filename>`
- Τα αρχεία μπορεί να είναι και σε υποκατάλογο





# Προσθήκη σε στιγμιότυπο 2

---

- Για απλούστευση κάντε και τα δύο βήματα μαζί, όταν θέλετε να κρατήσετε στιγμιότυπο:
  - `git add <filenames>`
  - `git commit`
- Η εντολή `git commit` ξεκινά έναν editor ώστε να γράψετε ένα σύντομο κείμενο που εξηγεί τι περιλαμβάνει το `commit`
  - εναλλακτικά: `git commit -m "το μήνυμα"`
- Τα κείμενα αυτά είναι σημαντικά
  - πρέπει να μπορεί κανείς να καταλαβαίνει τι κάνατε (και γιατί)
  - φανταστείτε ότι το γράφετε για κάποιον άλλο



# Προβολή των διαφορών

---

- Εντολή `git diff` δείχνει διαφορές
  - πολύ χρήσιμη σε συνδιασμό με την `git status`
- `git diff`
  - αλλαγές μεταξύ `staged` (μετά από `git add`) και `working directory`
- `git diff --cached`
  - αλλαγές μεταξύ τρέχοντος στιγμιότυπου (`HEAD`) και `staged`
- `git diff HEAD`
  - αλλαγές μεταξύ τρέχοντος στιγμιότυπου και `working directory`



# Απομακρυσμένο αποθετήριο

---

- Θυμηθείτε: το τοπικό αποθετήριο κλωνοποιήθηκε από ένα αποθετήριο του GitHub
  - το αποθετήριο του GitHub ονομάζεται απομακρυσμένο, **remote**
- Το remote repo έχει όνομα
  - Συνηθισμένο όνομα: origin
  - Εντολή **git remote** αναφέρει το όνομα του remote repo
    - **git remote -v** δίνει περισσότερες πληροφορίες
- Για παράδοση ασκήσεων
  - ενημερώνετε το αποκρυσμένο αποθετήριο με τις αλλαγές που έγιναν στο τοπικό
  - η πράξη λέγεται προώθηση **push**



# Πρωώθηση

---

- Το remote ενημερώνεται για όλα τα νέα στιγμιότυπα που δημιουργήθηκαν από την προηγούμενη ενημέρωση
- Προσοχή: αν στο working directory υπάρχουν **αλλαγές σε αρχεία, αυτές δεν προωθούνται**
  - μόνο commits προωθούνται
- Εντολή
  - Την πρώτη φορά: `git push -u origin master`
    - ώστε μετά το `git push` να αρκεί
  - Το origin είναι το όνομα του remote
  - Το **master** είναι το όνομα του παρακλαδιού που θέλετε να προωθήσετε. Το master είναι το «κύριο» παρακλάδι. Αναφέρεται και ως trunk (κορμός) σε διάφορα κείμενα



# Σύνοψη: git για το μάθημα

Στην αρχή (για κάθε τοπικό αποθετήριο):

- `git clone <private_repo_URL>`

βλ εξήγηση  
παρακάτω

Για κάθε καινούρια άσκηση τα ελάχιστα βήματα είναι:

- `git remote add labXX_starter <lab_starter_URL>`
- `git fetch labXX_starter`
- `git merge labXX_starter/master -m "Fetched labXX init files"`

- Θα δημιουργηθεί ο κατάλογος labXX
- Κάνετε αλλαγές αρχείων σχετικών με την άσκηση
  - λύση της άσκησης
- `git add <αρχεία που τροποποιήθηκαν, προστέθηκαν>`
- `git commit -m "μήνυμα"`
- `git push origin master`
  - ή `git push`, αν έχετε κάνει `git push -u origin master` την πρώτη φορά
- **Αναλυτικότερα στο Piazza <https://piazza.com/class/ijd4i19xaic3s8?cid=10>**



# Πιθανά σενάρια

---

- Σ1: Δουλεύοντας με 2 τοπικά αποθετήρια
  - π.χ. και στο εργαστήριο και στο σπίτι
- Σ2: Αναίρεση αλλαγών
  - αν κάτι πάει στραβά, πώς επαναφέρουμε ένα παλιότερο στιγμιότυπο
- Σ3: Αλλαγή ονόματος, διαγραφή αρχείου



# Σ1: Δύο τοπικά αποθετήρια

---

- Π.χ. στο εργαστήριο και στο σπίτι
- Δημιουργία 2 κλώνων
- Στο εργαστήριο, την πρώτη φορά
  - git clone, add, commit, push
  - ώστε όλες οι αλλαγές να προωθηθούν στο GitHub
- Στο σπίτι (υποθέτω υπάρχει ήδη κλώνος)
  - git pull – ώστε οι αλλαγές να «έρθουν» τοπικά
  - στο τέλος: git add, commit, push
- Μετά, πάντα ξεκινάτε με git pull



# Σ1: Πιθανά προβλήματα

---

- Αν ξεχάσετε να κάνετε push τις τελευταίες αλλαγές πριν φύγετε από το σπίτι
  - δεν υπάρχει τρόπος να τις πάρετε!
  - (εκτός αν μπορείτε να στήσετε ένα git server στο σπίτι, ...)
- Αν ξεχάσετε να κάνετε pull ώστε να έχετε την πιο πρόσφατη έκδοση
  - και κάνετε commit
    - ένα push θα αποτύχει γιατί το απομακρυσμένο αποθετήριο έχει αλλαγές που δεν έχετε τοπικά
    - πρέπει πρώτα να φέρετε και να συγχωνέψετε(**merge**) τις αλλαγές
  - αν δεν κάνατε commit,
    - το pull θα αποτύχει και θα σας ζητηθεί να κάνετε commit και μετά merge, όπως παραπάνω





# Σ1: Συνδιασμός αλλαγών

---

- Εντολή git pull
  - προσπαθεί να συνδιάσει τις αλλαγές αυτόματα,
    - προσθήκες νέων αρχείων δεν δημιουργούν προβλήματα
  - αλλά συχνά αποτυχαίνει
    - αλλαγές στο ίδιο αρχείο απαιτούν την παρέμβαση του χρήστη
  - Τα προβλήματα ονομάζονται **conflicts**
  - Θα δείτε ποιά αρχεία έχουν conflicts στην απάντηση της pull
- Επίλυση συγκρούσεων (συγχώνευση)
  - ανοίξτε τα αρχεία με ένα editor και θα δείτε τις αλλαγές
  - διορθώστε το πρόβλημα με το χέρι, σε όλα τα αρχεία
  - δημιουργείστε ένα νέο στιγμιότυπο με τις αλλαγές
    - git add, commit
  - προώθηση στο GitHub με git push



# remote add, fetch, merge

---

Στη σύνοψη git για το μάθημα, είδαμε 3 περιέργες εντολές git:

```
1. git remote add labXX_starter <lab_starter_URL>
2. git fetch labXX_starter
3. git merge labXX_starter/master -m "Fetched labXX init files"
```

1. Προσθήκη απομακρ. αποθετηρίου με όνομα labXX\_starter
  - μπορούμε να έχουμε πολλά remotes...
2. Προσκόμηση του τελευταίου στιγμιότυπου του labXX\_starter
  - δεν γίνονται όμως αλλαγές στα αρχεία του working directory
3. Συγχώνευση του remote με το working directory
  - θεωρητικά μπορεί να υπάρξουν συγκρούσεις
  - στο μάθημα, πάντα χωριστός κατάλογος



# Σ2: αναίρεση αλλαγών

---

- Πολλοί διαφορετικοί τρόποι ανάλογα με την κατάσταση
  - και το τρόπο εργασίας/προτίμηση
- Περιπτώσεις
  1. πριν τη προσωρινή προσθήκη (πριν κάνετε git add)
  2. μετά τη προσωρινή αλλά πριν τη τελική προσθήκη (πριν git commit)
  3. μετά τη τελική προσθήκη (μετά το git commit) αλλά πριν τη προώθηση σε απομακρυσμένο αποθετήριο
  4. μετά και από τη προώθηση σε απομακρυσμένο αποθετήριο
- Σημείωση: Αναίρεση δεν σημαίνει ότι η ώρα προσπάθειας αρχείων θα επανέλθει. Τα περιεχόμενα μόνο αλλάζουν.



# Π1: πρίν το git add

---

- Βεβαιωθείτε ότι πράγματι δεν έχετε κάνει git add
  - git status
  - **δεν** θα πρέπει να έχει Changes to be committed
- Δώστε την εντολή git checkout <όνομα αρχείου>
  - η εντολή φέρνει από το τρέχον στιγμιότυπο το αρχείο και το γράφει στο working directory



# Π2: πριν το git commit

---

- Βεβαιωθείτε για την κατάσταση του αποθετηρίου
  - git status
- Εντολές αναίρεσης:
  1. git reset
    - αφαιρεί από το staging area όλες τις αλλαγές από το τελευταίο στιγμιότυπο (αναιρεί τα προηγούμενα git add)
    - Αλλά το working directory δεν αλλάζει
  2. git checkout <ονόματα αρχείων>
    - «Φέρνει» τα αρχεία, όπως ήταν στο προηγούμενο στιγμιότυπο, στο working directory



# Π3: μετά το commit, πριν push

---

- Για αναίρεση του τελευταίου commit:
  - `git reset --hard HEAD~1`
  - οι αλλαγές στο `working dir` χάνονται και όλα τα αρχεία επανέρχονται όπως ήταν στο στιγμιότυπο.
- Το `HEAD~1` σημαίνει ένα commit πριν
  - θα μπορούσαμε να έχουμε `~2` κλπ.
  - μπορούμε επίσης να αναφερθούμε σε ένα commit με το hash του
- Δεν είναι καλή ιδέα για κώδικα που έχουμε κάνει push σε απομακρυσμένο αποθετήριο και είναι ορατός από άλλους γιατί τους μπερδεύει



# Π4: μετά το push

---

- Στιγμιότυπα που έγιναν push σε απομακρυσμένο αποθετήριο
  - έχουν γίνει ορατά και χρησιμοποιηθεί (pull) από άλλους
  - αν ξαναγυρίσουμε σε παλιότερο στιγμιότυπο, οι άλλοι χρήστες θα μπερδευτούν
- Καλύτερα να δημιουργηθεί ένα νέο στιγμιότυπο
  - που στην ουσία αναιρεί τις αλλαγές
- Εντολή git revert HEAD
  - ξεκινάει editor με ένα έτοιμο μήνυμα. Μπορείτε να το αλλάξετε
  - το working dir επανέρχεται στην κατάσταση του στιγμιότυπου
  - για αναίρεση πιο πίσω, HEAD~1 κλπ.
- Μετά, git push
  - ώστε η αλλαγή να περάσει στο απομακρυσμένο αποθετήριο



# Σ3: μετονομασία, διαγραφή

---

- Διαγραφή αρχείου
  - διαγράφεται από το working directory, αλλά όχι από το στιγμιότυπο
  - διαγραφή και από το working directory και από το (επόμενο) στιγμιότυπο: `git rm <file_name>`
- Μετονομασία/μετακίνηση αρχείου
  - παρόμοια προβλήματα με τη διαγραφή
  - εντολή `git mv`





# Πιο προχωρημένη χρήση

---

## Παρακλάδια (branches)

- Δημιουργία νέου branch
- Αλλαγή τρέχοντος branch
- Συγχώνευση παρακλαδιών
- Ενημέρωση remote για τα παρακλάδια



# Γιατί παρακλάδια

---

- Συχνές δοκιμές κατά την ανάπτυξη κώδικα
  - πώς θα οργανωθεί αυτό καλά;
- Όλα τα version control systems υποστηρίζουν παρακλάδια
  - το git φημίζεται ότι το κάνει πολύ εύκολα



# Δημιουργία

---

- Εντολή:
  - `git checkout -b <branch_name>`
- Δημιουργεί το παρακλάδι και αλλάζει το τρέχον παρακλάδι σε αυτό (στο καινούριο)
- Η εντολή `git status` στην πρώτη γραμμή αναφέρει σε ποιο παρακλάδι βρισκόμαστε κάθε στιγμή



# Αλλαγή τρέχοντος branch

---

- Για να δούμε ποιά παρακλάδια υπάρχουν:
  - εντολή: `git branch`
- Αλλαγή τρέχοντος branch
  - Εντολή: `git checkout <branch_name>`
- Αλλαγμένα αρχεία στο `working directory`, δεν χάνονται!



# Συγχώνευση παρακλαδιών

---

- Όπως συγχωνεύσαμε τοπικό με απομακρυσμένο αποθετήριο (merge, pull), έτσι γίνεται και σε branches
- Στο τρέχον παρακλαδι, έστω master, μπορούμε να συγχωνεύσουμε τις αλλαγές που έκανε το παρακλάδι testbranch
  - git merge testbranch
  - Φυσικά μπορεί να υπάρξουν συγκρούσεις που πρέπει να λυθούν
- Προσοχή: έχει σημασία ποιό παρακλάδι συγχωνεύεται σε ποιό!
  - το master στο testbranch ή το testbranch στο master?
  - και τα δύο έχουν νόημα για διαφορετικούς λόγους



# Ενημέρωση remote για branches

---

- Συνήθως τα παρακλάδια είναι για «προσωπική χρήση»
  - δοκιμάζει κανείς παραλλαγές στο τοπικό του αποθετήριο
  - πετάει ό,τι δεν δουλεύει και τις χρήσιμες αλλαγές τις συγχωνεύει με το κύριο παρακλάδι (master)
  - μετά στέλνει τις «τελικές» αλλαγές (του master) στο απομακρυσμένο αποθετήριο
- Αν στα παρακλάδια δουλεύουν και άλλοι πρέπει να φαίνονται στο απομακρυσμένο αποθετήριο
  - εντολή: `git push <remote> <branch>`



# Σύνοψη – git, GitHub

---

- Τί είναι;
  - γιατί στο μάθημα αυτό;
- Βασικές έννοιες
  - στιγμιότυπο, αποθετήριο, κατάλογος εργασίας
  - κλωνοποίηση, προώθηση, ενσωμάτωση αλλαγών
- Βασικές εντολές
- Workflow για τις ασκήσεις
- Πιθανά σενάρια/προβλήματα
- Παρακλάδια



# Επόμενο μάθημα

---

MIPS assembly –

απλές εντολές με καταχωρητές και  
εντολές προσπέλασης μνήμης

