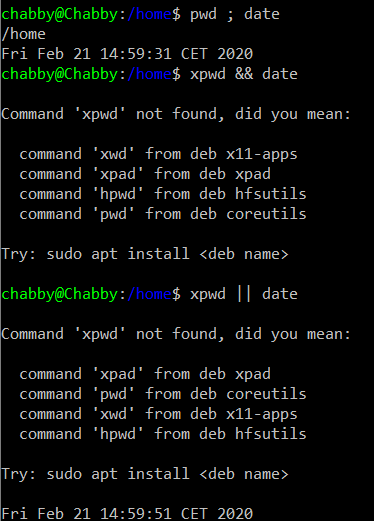
**Sistemi Operativi: Laboratorio**

**Progetto**:

* **Sistema Operativo** da usare è da definire, ma probabilmente verrà usato **Ubuntu v18.04**.
* Il progetto consiste nel realizzare un **programma in C** che soddisfa dei requisiti (verranno specificati successivamente).
* I requisiti che il programma deve soddisfare saranno in una forma discorsiva (possiamo realizzarlo come vogliamo, basta che faccia quello che deve fare).
* Se il programma compila e funziona non è detto che si passi (guardano anche il modo in cui è stato realizzato) come se non compila o ci sono problemi, non è detto che non si passi il progetto:
  + C’è una **fase di discussione** (ci si può dividere i compiti, ma tutti devono sapere tutto del progetto).

**Comandi**:

* Concatenazione di comandi:
  + **pwd ; date**:
    - con il ‘**;**’ vengono concatenati i comandi (viene eseguito pwd e una volta terminata la sua esecuzione, viene eseguito il comando date).
  + **pwd && date**:
    - anche in questo caso si ha una “*concatenazione*” di comandi, ma si ha un comportamento diverso rispetto a quello descritto sopra.
    - Visto che concatena con un **AND** logico, se il comando **pwd** viene eseguito con successo, allora anche il comando date viene eseguito.
    - Se il comando **pwd** non dovesse essere eseguito, neanche date viene eseguito (F && X = F)
  + **pwd || date**:
    - caso simile a quello dell’AND solo che si ha un **OR** logico tra i due comandi.
    - Se **pwd** non viene eseguito, **date** viene eseguito lo stesso
* Comando per contare righe/parole:
  + **wc** (word count)
    - **ls /home | wc**:
      * **ls /home** genera dei dati (in questo caso elenca tutti gli elementi presenti nella cartella **/home**) e li invia sul canale di uscita che conosce (il comando **ls** non sa dove vanno i dati, lui sa solo che i dati in uscita vanno mandati in quel canale).
      * La **bash** pesca questi dati e li mostra a video.
      * Il carattere ‘**|**’ fa si che i dati che il comando **ls** genera, invece che mandarli sulla rotta di default, li mandi al comando **wc**.
      * **wc** riceve questi dati, li elabora e anche lui manda i dati in uscita sul suo canale di output di default (poi la **bash** mostrerà questi dati).
      * Notare che l’output che viene mostrato è l’output dell’ultimo comando eseguito (non sempre si può inoltrare i dati generati da un comando ad un altro comando; quest’ultimo deve accettare dati in input).



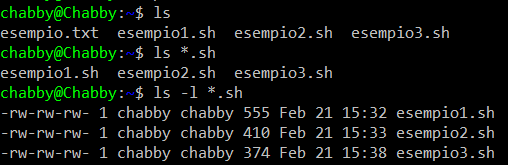
* + - **wc < tmp/esempio.txt**:
      * viene letto il file **esempio.txt** e tutti i dati presi dal file vengono passati in input al comando **wc** (tramite una **pipe** vengono passati questi dati letti).
      * poi **wc** usando questi dati, genera l’output e lo inoltra sul suo canale di output (che verrà usato dalla **bash** per mostrare i dati a video).
      * il simbolo ‘**<**’ viene usato per dire “prendi i dati dal file **esempio.txt**” e passali in input al comando **wc**.



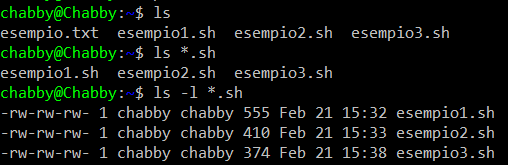
* Note:
  + I comandi hanno 3 canali:
    - **Input**: canale di entrata (tastiera)
    - **Output**: canale di uscita (video)
    - **Error**: usando questo canale, posso differenziare errori dai dati corretti
* Comando which:
  + Il comando **which** mi restituisce il percorso dove è salvato il programma che sto cercando di / voglio eseguire(**path**).



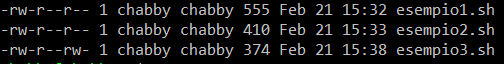
* Bash:
  + La **bash** è un programma che apre un **prompt** e aspetta che l’utente scriva “qualche cosa”.
  + Quel “qualcosa” viene interpretato come comando e viene eseguito (se non riconosce il comando, la **bash** restituisce un errore).
  + Quando scrivo in comando, il programma **bash** lo cerca nella sua cartella e nei posti che conosce (se non è un comando interno) e lo esegue.
* Comando per elencare informazioni su **file** ed il contenuto di **directory**:
  + **ls**:
    - mostra informazioni ed il contenuto di file e cartelle



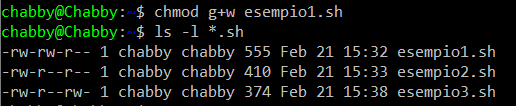
* + **ls \*.sh**:
    - elenca tutti i file che hanno come estensione .sh



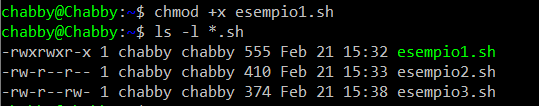
* + **ls -l \*.sh**:
    - restituisce i dati relativi ai file: tipo di file, nome del possessore, nome del gruppo, dimensione, ultima modifica e i permessi associati a quel file, divisi in 3 gruppi (proprietario, domain user e user):
      * **rw**: read & write (posso sia leggere che scrivere)
      * **r**: read (posso solo leggere)



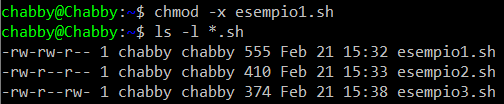
* Comando per modificare i **permessi** per un file:
  + **Opzioni**:
    - **a**: all | **g**: group (domain user) | **u**: user | **+**: aggiungo permesso | **-**: tolgo permesso
    - **r**: read | **w**: read | **x**: execution
  + **chmod g+w esempio1.sh**:
    - modifico i permessi del file esempio1.sh e modifico i permessi del domain user, che ora possono anche scrivere (oltre che leggere).



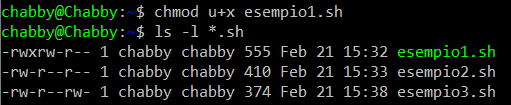
* + **chmod +x esempio1.sh**:
    - aggiungo il permesso di esecuzione (**x**) per tutti gli utenti per il file **esempio1.sh**
    - sto dichiarando che il file **esempio1.sh** deve essere trattato come un eseguibile.



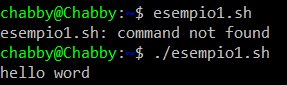
* + **chmod -x esempio1.sh**:
    - tolgo il permesso di esecuzione per il file **esempio1.sh**



* + **chmod u+x esempio1.sh**:
    - modifico i permessi solo per lo user (chi ha creato il file) e solo lui può eseguire il file **esempio.sh**



* Comando per eseguire:
  + Se provo ad eseguire il file, ottengo un errore, in quanto quando **bash** esegue un comando, non controlla mai la cartella corrente dove è situato.
  + Per far sì che venga eseguito devo scrivere l’intero percorso del file.
  + **./esempio1.sh:**



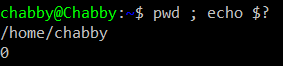
* + **bash esempio1.sh**:
    - passo a bash il file che lo esegue



* Comando per operazioni matematiche:
  + **bc**:
    - accetta **espressioni matematiche** e le elabora.
    - prende in input l’espressione matematica, elabora e fa i suoi calcoli e successivamente stampa il risultato.
  + **echo ‘1.12 + 2.02’ | bc**:
    - prima di stampare a video i dati, questi vengono passati a **bc** in input (operatore ‘**|**’) che esegue la somma, restituisce il valore e questo viene stampato a video tramite il comando **echo**.



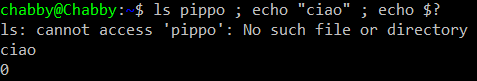
* Codice di uscita:
  + Tutti i comandi restituiscono un **codice di uscita (0-255)** al chiamante (al processo **bash**).
  + Ogni volta che viene eseguito un comando, **$?** viene **inizializzato** con il **codice di uscita** dell’**ultimo** **comando eseguito**.
  + **pwd ; echo $?** restuisce **0**, in quanto il comando **pwd** viene eseguito e ritorna al chiamante **0**, cioè “l’esecuzione è avvenuta con successo”:



* + **ls pippo ; echo $?**:
    - mi restituisce un errore, in quanto il comando pippo non esiste (**bash** non lo conosce e non lo ha tra i programmi) e ho un codice di errore **2**.
    - **ls** quando non trova un file ha come **codice di errore 2**.

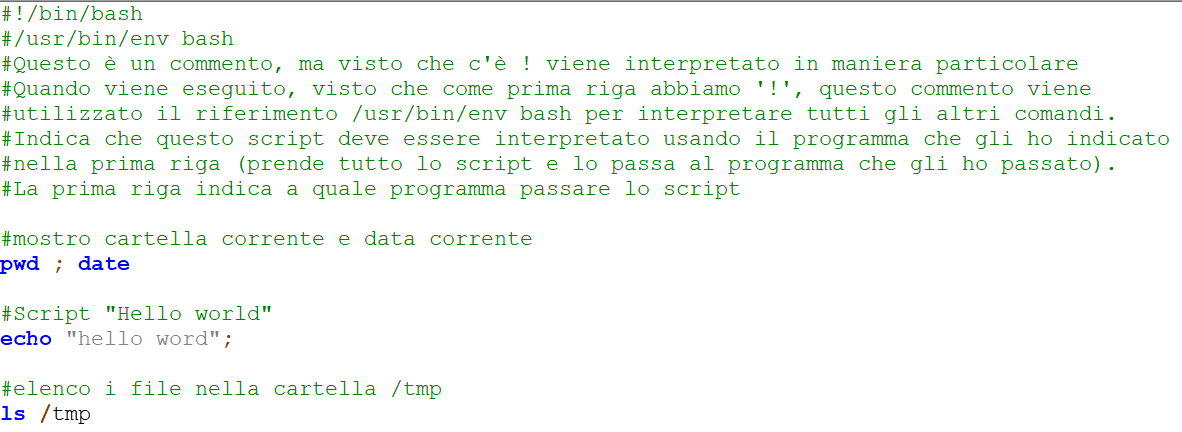


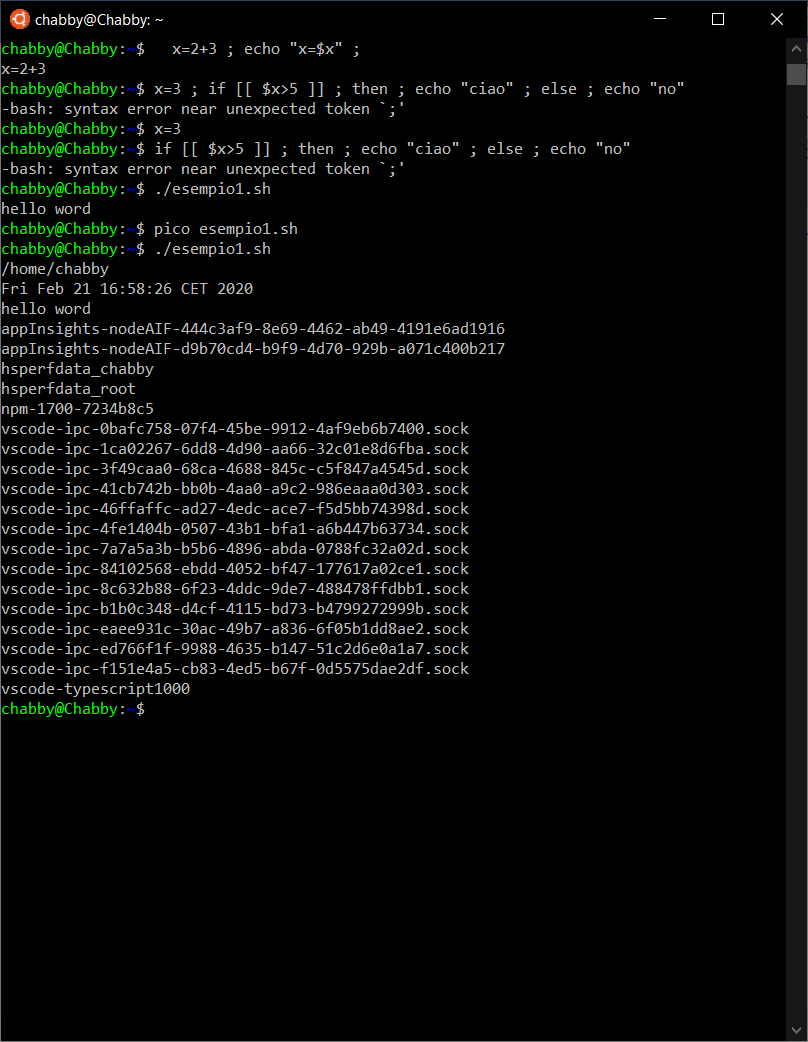
* + Visto che **$?** contiene sempre il **codice di uscita dell’ultimo comando eseguito**, se io eseguo un comando (che non viene eseguito per qualsiasi problema) e poi faccio altro, nella variabile **$? non trovo più il codice di uscita di quel comando**, ma trovo il **codice di uscita del comando eseguito appena prima di $?**:



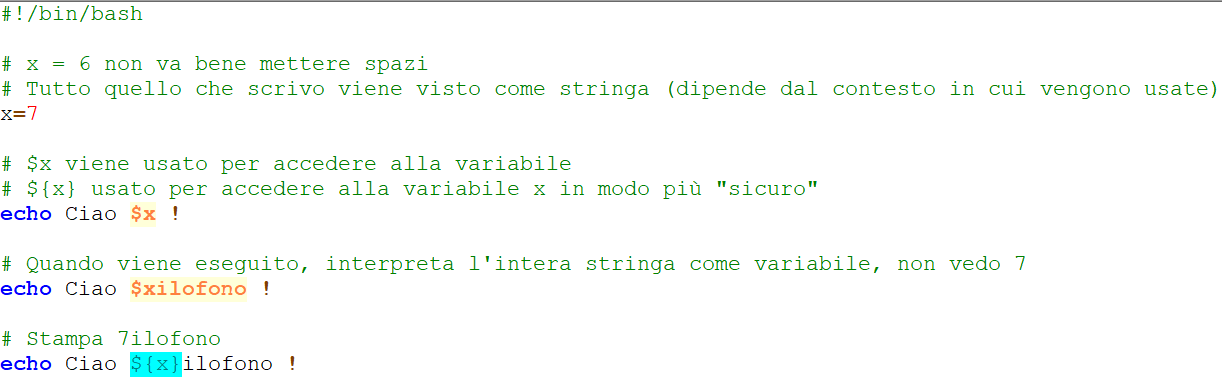
* Comandi per il confronto:
  + Vedi slide (sito docente o drive)

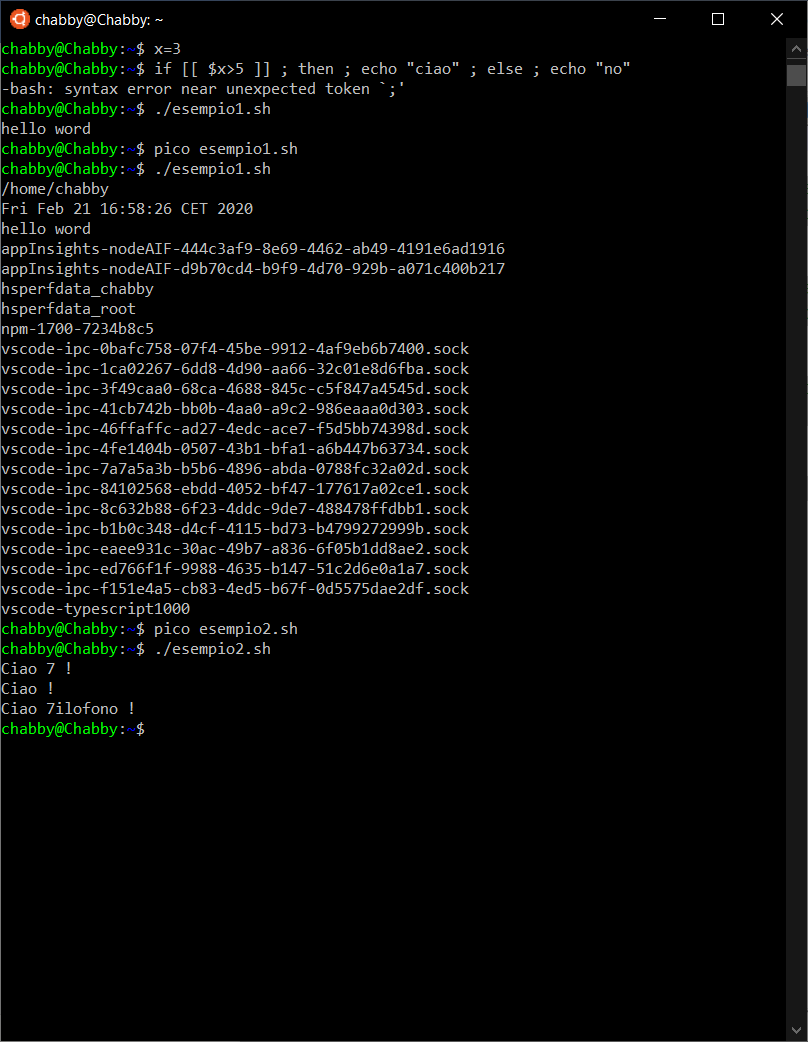
Esempio 1:



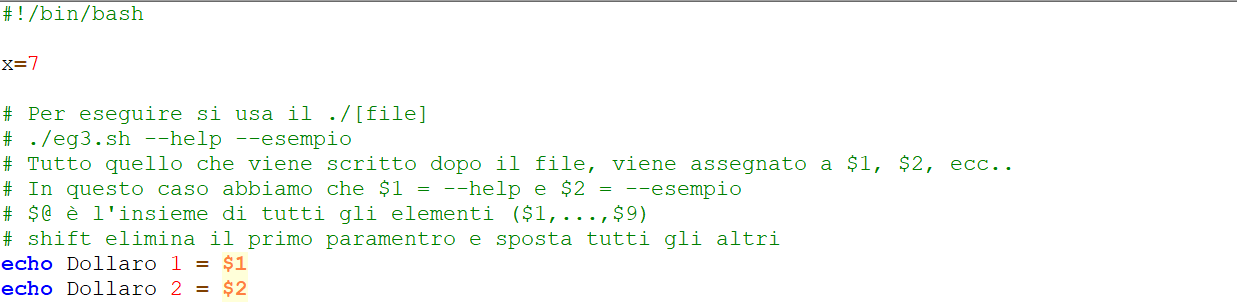


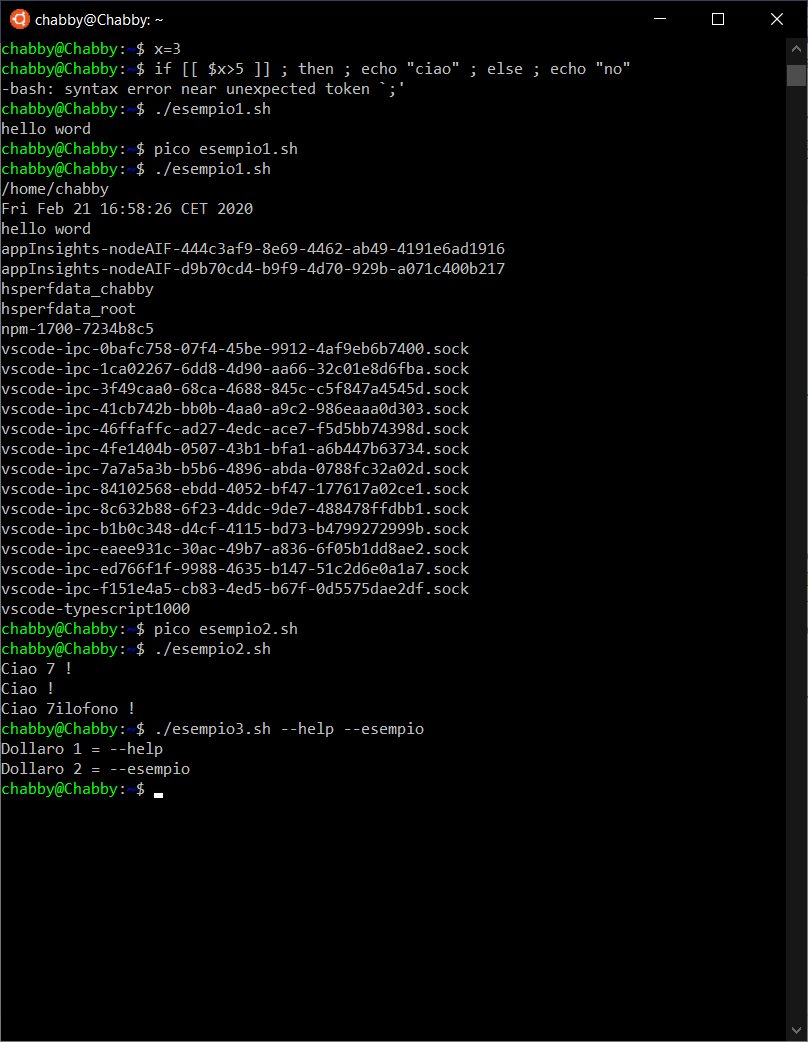
Esempio 2:



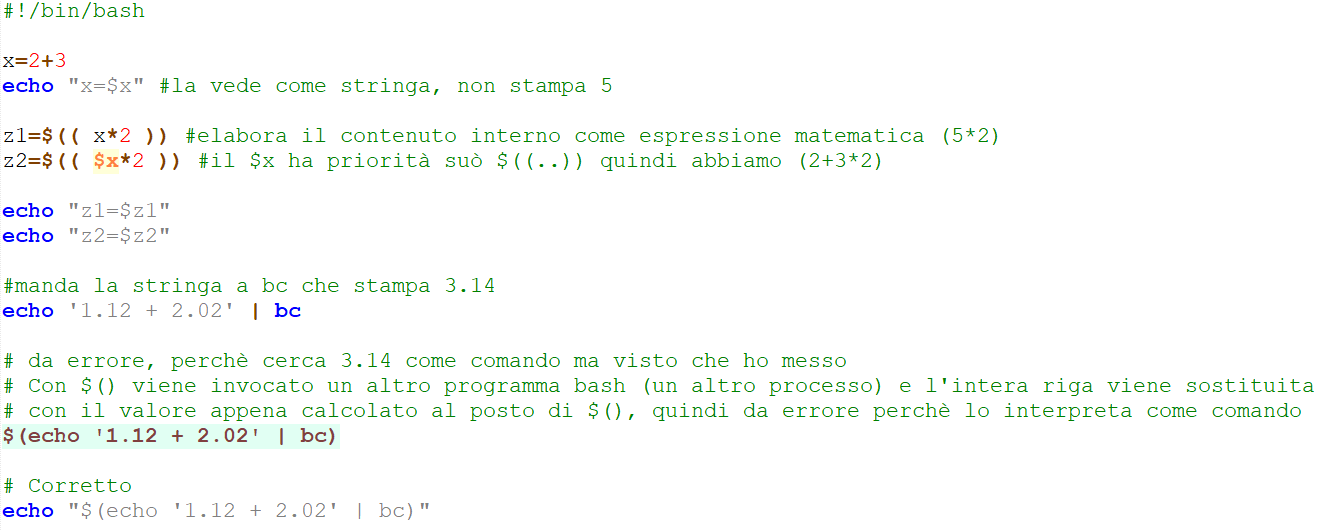


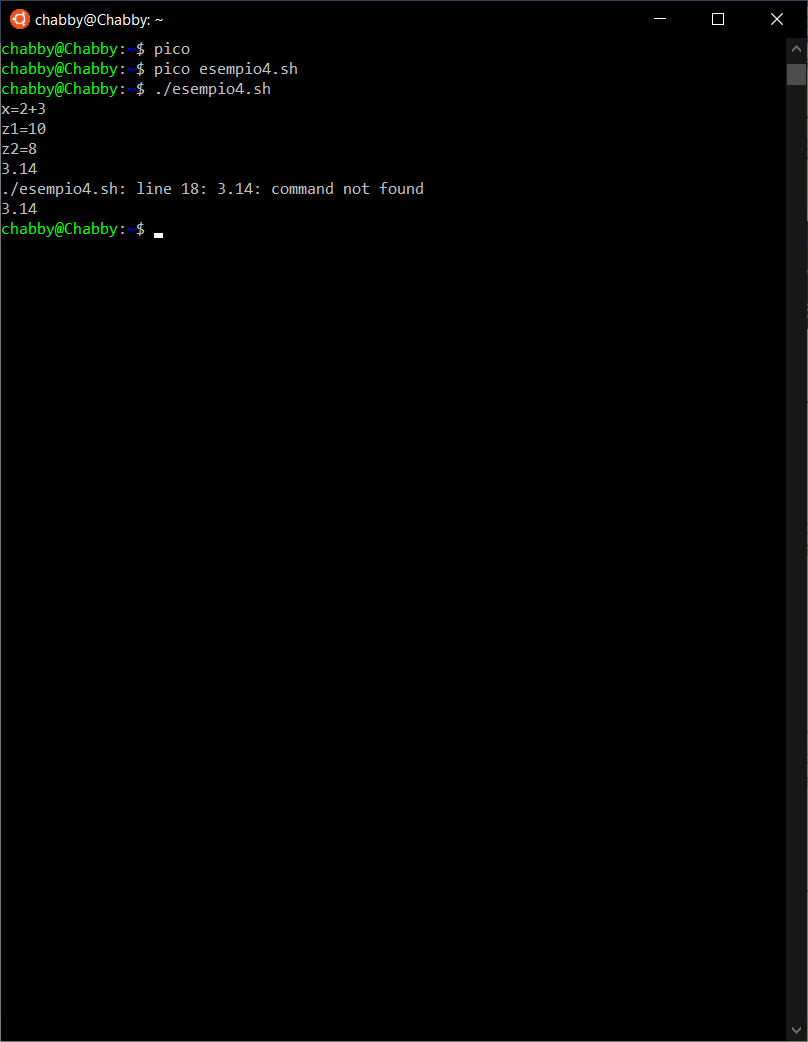
Esempio 3:



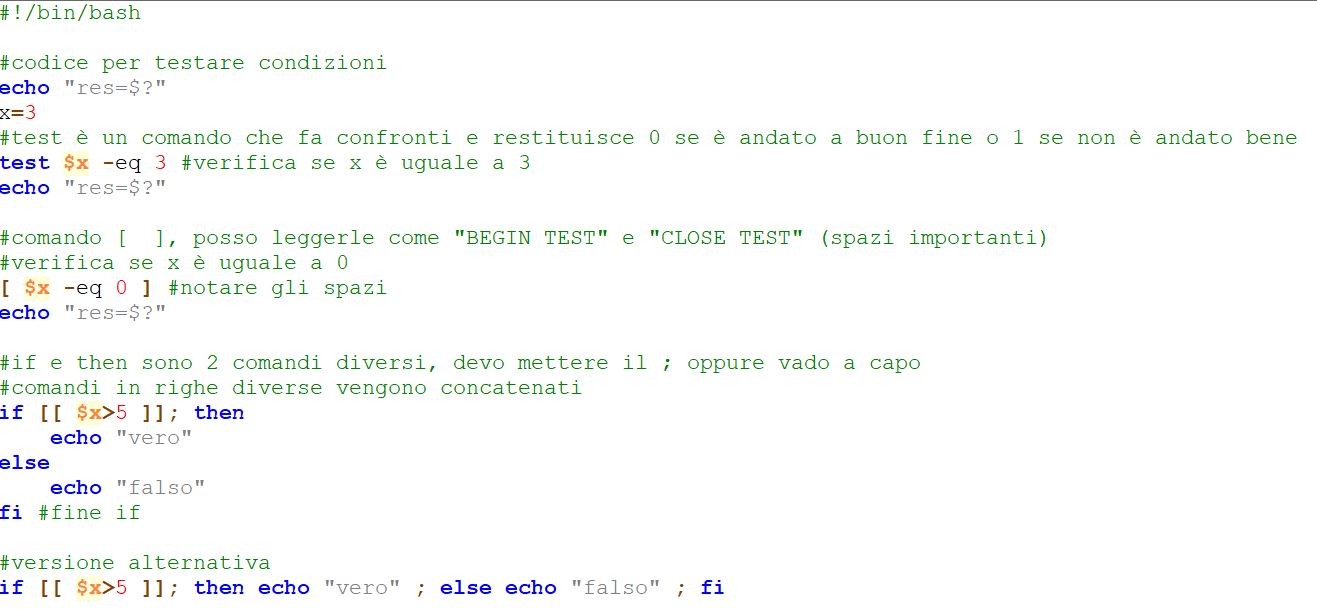


Esempio 4:





Esempio 5:





Esempio 6:

