A cura di:
Lorenzo Aliotta
Riccardo Dal Seno
Teresa de Jesus Fernandes

RELAZIONE DI LABORATORIO: MICROBASH (SETI)

line 461, getcwd (dentro main)

scopo: verificare il funzionamento della funzione getcwd
situazione iniziale: -
linea inviata alla microbash: - (viene mandato in esecuzione il programma)
risultato atteso: stampa del prompt della microbash con indicata la cartella di lavoro corrente
Nota: in condizioni particolari (i.e. out of memory, indirizzo da stampare di dimensioni eccessive, …), la funzione fallisce e il programma si arresta stampando l'errore

```
linuxsofia@GranFausto:~/microbash$ ./microbash
/home/linuxsofia/microbash $
```

//

line 178, dentro parse_cmd

scopo: verificare che la microbash espanda le variabili d'ambiente esistenti, indicate con il simbolo "$" seguito dal nome della variabile (senza spazi tra i due)
situazione iniziale: la variabile d'ambiente PATH è definita nell'ambito della microbash
linea inviata alla microbash: echo $PATH
risultato atteso: stampa del valore della variabile PATH
Nota: questi risultati si ottengono solo con variabili d'ambiente predefinite, in quanto non è possibile definirne di nuove tramite la microbash

```
/home/linuxsofia/microbash $ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Program
Data/Oracle/Java/javapath:/mnt/c/Program Files (x86)/Intel/iCLS Client/:/mnt/c/Program Files/Intel/iCLS Client/:/mnt/c/W
indows/system32:/mnt/c/Windows:/mnt/c/Windows/System32/Wbem:/mnt/c/Windows/System32/WindowsPowerShell/v1.0/:/mnt/c/Windo
ws/System32/OpenSSH/:/mnt/c/Program Files (x86)/Intel/Intel(R) Management Engine Components/DAL:/mnt/c/Program Files/Int
el/Intel(R) Management Engine Components/DAL:/mnt/c/Program Files (x86)/Intel/Intel(R) Management Engine Components/IPT:
/mnt/c/Program Files/Intel/Intel(R) Management Engine Components/IPT:/mnt/c/Program Files/dotnet/:/mnt/c/Program Files/C
Make/bin:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v
1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files/MATLAB/R2023b/bin:/mnt/c/Users/Sofia/AppData/Local/Programs/P
ython/Python38-32/Scripts/:/mnt/c/Users/Sofia/AppData/Local/Programs/Python/Python38-32/:/mnt/c/Users/Sofia/AppData/Loca
l/Microsoft/WindowsApps:/mnt/c/Users/Sofia/.dotnet/tools:/snap/bin:/home/linuxsofia/.dotnet/tools
/home/linuxsofia/microbash $
```

//

scopo: verificare che la microbash "espanda" le variabili d'ambiente non esistenti, sostituendole con una stringa vuota nella linea di comando
situazione iniziale: la variabile d'ambiente ZENOBIA non è definita
linea inviata alla microbash: echo $ZENOBIA
risultato atteso: stampa della stringa vuota

```
/home/linuxsofia/microbash $ echo $ZENOBIA

/home/linuxsofia/microbash $
```

//

scopo: verificare che la microbash stampi un errore se il simbolo $ non è seguito dal nome di una variabile (spazio vuoto o fine della linea di comando)
situazione iniziale: -
linea inviata alla microbash: echo $
risultato atteso: Parsing error: no variable name specified

```
/home/linuxsofia/microbash $ echo $
Parsing error: no variable name specified
/home/linuxsofia/microbash $
```

//

line 233, dentro check_redirections
scopo: verificare la correttezza delle redirezioni in I/O in una linea di comando
situazione iniziale: il file microbash.c esiste, mentre il file out.txt se esiste viene sovrascritto, se non esiste viene creato un nuovo file nominato “out.txt”
linea inviata alla microbash: cat <microbash.c | grep main | wc -l >out.txt
risultato atteso: non viene stampato nessun errore (in particolare, dopo l’esecuzione di questa linea di comando, il file out.txt conterrà il numero di volte in cui la stringa “main” compare nel codice sorgente)

```
/home/linuxsofia/microbash $ cat <microbash.c | grep main | wc -l >out.txt
/home/linuxsofia/microbash $ cat <out.txt
1
/home/linuxsofia/microbash $
```

//

scopo: verificare che, se nella linea di comando sono specificate redirezioni errate, venga stampato un errore per notificarle
situazione iniziale: -
linea inviata alla microbash: cat <in.txt >out.txt | echo >out.txt EOF
risultato atteso: Stampa dell’errore "Output redirection in a non-last command" e interruzione dell’esecuzione della linea di comando
Nota: lo stesso comportamento si osserva per redirezioni multiple in input (in quel caso l’errore sarà: "Input redirection in a non-first command")

```
/home/linuxsofia/microbash $ cat <in.txt >out.txt | echo >out.txt EOF
Output redirection in a non-last command
```

//

line 262, dentro check_cd:

scopo: cambiare la current working directory
situazione iniziale: nella cartella microbash è presente la cartella dir
linea inviata alla microbash: cd dir
risultato atteso: la cartella di lavoro corrente viene aggiornata con l'argomento della funzione cd

```
/home/linuxsofia/microbash $ mkdir dir
/home/linuxsofia/microbash $ cd dir
/home/linuxsofia/microbash/dir $
```

//

scopo: verificare che la presenza di cd in una successione di comandi venga segnalata come errata
situazione iniziale: -
linea inviata alla microbash: rm file.txt | cd .. | rmdir dir
risultato atteso: viene segnalato un errore ("cd is not the only command") e non viene eseguita la linea di comando

```
/home/linuxsofia/microbash $ rm file.txt | cd .. | rmdir dir
cd is not the only command
/home/linuxsofia/microbash $
```

//

scopo: verificare che la presenza di una redirezione (in input o in output) sul comando cd venga segnalata come errata
situazione iniziale: -
linea inviata alla microbash: cd <in.txt
risultato atteso: viene segnalato un errore ("cd cannot have I/O redirections") e non viene eseguita la linea di comando
Nota: -

```
/home/linuxsofia/microbash $ cd <in.txt
cd cannot have I/O redirections
```

//

scopo: verificare che venga segnalato un errore se al comando cd viene fornito un numero di argomenti diverso da 1
situazione iniziale: nella cartella microbash sono presenti le cartelle dir e dir2
linea inviata alla microbash: cd dir dir2
risultato atteso: viene segnalato un errore ("cd must have one argument") e non viene eseguita la linea di comando
nota: check_cd si limita a verificare il numero di argomenti passati a cd, non considerando il tipo di argomenti e la loro esistenza

```
/home/linuxsofia/microbash $ cd dir dir2
cd must have one argument
/home/linuxsofia/microbash $
```

//

line 393, in execute_line:

scopo: eseguire il comando open su due file esistenti e accessibili in apertura e scrittura
situazione iniziale: i file in.txt e out.txt esistono e sono accessibili; in particolare, in.txt contiene la stringa “hello world”
linea inviata alla microbash: cat <in.txt >out.txt …
risultato atteso: la linea di comando è corretta e viene correttamente eseguito il comando
nota: non è rilevante il contenuto del file out.txt in quanto verrà comunque sovrascritto; se il file out.txt non esistesse, verrebbe creato

```
/home/linuxsofia/microbash $ cat <in.txt >out.txt
/home/linuxsofia/microbash $ cat <out.txt
hello world
/home/linuxsofia/microbash $
```

//

scopo: eseguire il comando open su un file non esistente in apertura
situazione iniziale: il file input.txt non è presente nella directory corrente
linea inviata alla microbash: cat <input.txt
risultato atteso: viene segnalato un errore (“cannot open input file: no such file or directory”) e non viene eseguita la linea di comando

```
/home/linuxsofia/microbash $ cat <input.txt
cannot open input file: No such file or directory
/home/linuxsofia/microbash $
```

//

scopo: eseguire il comando open su un file non accessibile in scrittura
situazione iniziale: il file outPrivate.txt non è accessibile in lettura e/o scrittura, mentre il file in.txt è completamente accessibile
linea inviata alla microbash: cat <in.txt >outPrivate.txt
risultato atteso: viene segnalato un errore (“cannot open output file: Permission denied”) e non viene eseguita la linea di comando

```
----------  1 linuxsofia linuxsofia    16 Nov 21 18:41 outPrivate.txt
/home/linuxsofia/microbash $ cat <in.txt >outPrivate.txt
cannot open output file: Permission denied
```

//

line 340, in run_child:

scopo: verificare la chiamata ad un programma non esistente nella microbash
situazione iniziale: il programma “dog” non esiste nelle directory indicate da PATH
linea inviata alla microbash: dog
risultato atteso: viene segnalato l’errore indicando il PID del processo figlio arrestato, con il valore di uscita relativo
nota: quando la funzione execvp fallisce, la wait_for_children (line 282) riconosce che il processo figlio è uscito con valore di errore e si occupa di stampare il messaggio di errore relativo

```
linuxsofia@GranFausto:~/microbash$ ./microbash
/home/linuxsofia/microbash $ dog
error in exec: No such file or directory
process with PID = 1145 exited with status 1
/home/linuxsofia/microbash $
```

//

line 283, in wait_for_children:

scopo: verificare i dati relativi ad un processo, dopo che questo è stato alterato nel suo stato da parte di un segnale (in particolare, l’alterazione dello stato qui considerata è la terminazione)
situazione iniziale: sono presenti due terminali:
terminale 1: su questo viene eseguita la microbash tramite la quale viene eseguito il programma a.out, che avvia un ciclo infinito
terminale 2: su di esso viene mandato il segnale alla microbash per terminare il segnale
linea inviata alla microbash (terminale 1): ./a.out
linea inviata al terminale (terminale 2): kill 1684
risultato atteso: la microbash stampa il PID del processo figlio, assieme al numero del segnale che ne ha causato la terminazione
nota: per trovare l’argomento della funzione kill, si è utilizzato il comando top nel terminale 2 con il quale è stato possibile determinare il PID del processo avviato dalla microbash

```
 1452 linuxso+  20   0    5880   1592   1436 R  99.7   0.1  12:53.18 a.out
 1684 linuxso+  20   0    5880   1532   1380 R  99.7   0.1   0:12.62 a.out
    1 root      20   0    1952   1304   1108 S   0.0   0.1   0:05.14 init
    8 root      20   0    1812     92      0 S   0.0   0.0   0:01.22 init
```

```
/home/linuxsofia/microbash $ ./a.out
process with PID = 1684 changed state due to signal 15: Terminated
/home/linuxsofia/microbash $
```

```
linuxsofia@GranFausto:~$ kill 1684
linuxsofia@GranFausto:~$
```

//

Note aggiuntive

Alcune aree di codice non vengono effettivamente raggiunte dai test mostrati nella relazione; di seguito vengono elencate le linee corrispondenti e i motivi per cui non è stato possibile eseguire test specifici:

- line 313, in redirect;
  non è stato possibile testare le funzioni dup2 e close perché non sono riproducibili i casi in cui queste falliscono

- line 343, in run_child;
  non è stato possibile testare un caso in cui la fork fallisca, in quanto i casi di errore non dipendono dall'utente; ad esempio, fork fallisce quando non c'è più memoria nel sistema o se è stato raggiunto il limite di processi attivi contemporaneamente

- line 420, in execute_line;
  non è stato possibile testare un caso in cui pipe2 fallisca, in quanto i casi di errore non dipendono dall'utente; ad esempio, pipe2 fallisce quando non c'è più memoria nel sistema o non sono disponibili ulteriori file descriptor