Esame Sistemi Operativi - Operating Systems Exam
2024/02/27

Ex 01 (3.5 points)

Italiano

Si supponga di eseguire il seguente programma. Si indichino quali sono gli output prodotti ammissibili. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose to run the following program. Indicate the possible admissible outputs. Note that wrong answers imply a penalty in the final score.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "pthread.h"
#include "semaphore.h"

sem_t s1, s2, me1, me2;

static void *TA ();
static void *TB ();
static void *TC ();

int main (int argc, char **argv) {
  pthread_t th1, th2, th3;

  sem_init (&s1, 0, 1);
  sem_init (&s2, 0, 0);
  sem_init (&me1, 0, 1);
  sem_init (&me2, 0, 0);

  setbuf(stdout, 0);

  pthread_create (&th1, NULL, TA, NULL);
  pthread_create (&th2, NULL, TB, NULL);
  pthread_create (&th3, NULL, TC, NULL);

  pthread_exit(0);
}

static void *TA () {
  pthread_detach (pthread_self ());

  sem_wait (&me1);
  for (int i=0; i<2; i++) {
    sem_wait (&s1);
    printf ("A"); fflush (stdout);
    sem_post (&s1);
    if (i==1) {
      sem_post (&me1);
    }
  }
  sem_post (&s2);
```

```
  sem_wait (&me2);

  return 0;
}


static void *TB () {
  pthread_detach (pthread_self ());

  sem_wait (&me1);
  for (int i=0; i<2; i++) {
    sem_wait (&s1);
    printf ("B"); fflush (stdout);
    sem_post (&s1);
    if (i==1) {
      sem_post (&me1);
    }
  }
  sem_post (&s2);
  sem_wait (&me2);

  return 0;
}

static void *TC () {
  pthread_detach (pthread_self ());

  sem_wait (&s2);
  sem_wait (&s2);
  printf ("C\n");
  sem_post (&me2);
  sem_post (&me2);

  return 0;
}
```

Scegli una o più alternative. Choose one or more options.

1. ⬜ ABABC
2. ✅ BBAAC
3. ⬜ AAAAC
4. ✅ AABBC
5. ⬜ BBBBC
6. ⬜ BABAC

Ex 02 (5.0 points)

Italiano

Implementare un programma concorrente che, utilizzando il linguaggio C e i semafori della libreria Pthread, crei tre thread:

- Il primo thread è costituito da un ciclo indefinito; in ogni iterazione il thread attende un tempo casuale che varia tra 0 a 5 secondi, genera un atomo di Cloro (Cl) e attende che il terzo thread finisca il suo compito.
- Il secondo thread è costituito da un ciclo indefinito; in ogni iterazione il thread attende un tempo casuale che varia tra 0 a 5 secondi, genera un atomo di Sodio (Na) e attende che il terzo thread finisca il suo compito.

- Il terzo thread produce una molecola di Cloruro di Sodio (NaCl) non appena sono disponibili un atomo di Sodio (Na) e un atomo di Cloro (Cl). Una volta generata la molecola permette all'intero processo di ripetersi.

L'attesa casuale che varia tra 0 e 5 secondi può essere ottenuta con la funzione `sleep(rand()%6)`. Si simuli la generazione degli atomi di Na e Cl e della molecola NaCL visualizzando le stringhe "Na", "Cl" e "NaCl".
Il programma fornito dovrà essere completo e dovrà includere il main, i thread e tutte le variabili utili per la loro gestione.

English

Implement a program in concurrent C language using Pthreads and semaphores that creates three threads:

- The first thread loops indefinitely, and for each iteration, it generates an atom of Chlorine (Cl) in a time range varying from 0 to 5 seconds; then it waits for the finish of the task of the third thread.
- The second thread loops indefinitely, and for each iteration, it generates an atom of Sodium (Na) in a time range varying from 0 to 5 seconds; then it waits for the finish of the task of the third thread.
- The third thread produces a sodium chloride molecule (NaCl) whenever a Sodium (Na) atom and a Chlorine (Cl) atom are available. Once the molecule is generated, it allows the entire process to repeat/restart.

The random wait can be obtained with the following function `sleep(rand()%6)`.
Once a molecule of NaCl is created, the entire process restarts.
Simulate the generation of the Na and Cl atoms, and of the NaCl molecule by printing the strings "Na", "Cl" and "NaCl" when they are generated. The provided program must be complete, and it must include the main, the threads and all the variables needed to manage them.

Soluzione. Solution.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t sem_cl, sem_na, sem_nacl;


void *chlorine_thread(void *arg) {
  while (1) {
    sem_wait(&sem_cl);

    sleep(rand() % 6);
    printf("Cl\n");

    sem_post(&sem_nacl);
  }
  return NULL;
}


void *sodium_thread(void *arg) {
  while (1) {
    sem_wait(&sem_na);

    sleep(rand() % 6);
    printf("Na\n");

    sem_post(&sem_nacl);
  }
  return NULL;
}
```

```
void *molecule_thread(void *arg) {
  while (1) {
    sem_wait(&sem_nacl);
    sem_wait(&sem_nacl);

    printf("NaCl\n\n");

    sem_post(&sem_na);
    sem_post(&sem_cl);
  }
  return NULL;
}

nt main() {
  sem_init(&sem_cl, 0, 1);
  sem_init(&sem_na, 0, 1);
  sem_init(&sem_nacl, 0, 0);

  pthread_t th_cl, th_na, th_nacl;

  pthread_create(&th_cl, NULL, chlorine_thread, NULL);
  pthread_create(&th_na, NULL, sodium_thread, NULL);
  pthread_create(&th_nacl, NULL, molecule_thread, NULL);

  // Join threads (not necessary in this case because threads run indefinitely)
  pthread_join(th_cl, NULL);
  pthread_join(th_na, NULL);
  pthread_join(th_nacl, NULL);

  sem_destroy(&sem_cl);
  sem_destroy(&sem_na);
  sem_destroy(&sem_nacl);

  return 0;
}
```

Ex 03 (5.0 points)

Italiano

Uno script il cui nome è `filter.sh` riceve tramite linea di comando due interi (`n1` e `n2`) e un insieme di stringhe che rappresentano dei nomi di file:

`filter.sh n1 n2 file1 file2 file3 ...`

Si scriva uno script che:

- Verifica che il numero di parametri passati tramite linea di comando sia maggiore di tre, controlla che `n2` sia maggiore di `n1` (cioè, `n2>n1`), verifica che tutti i file indicati siano presenti nel file system.
- Nel caso una delle condizioni precedenti sia falsa lo script deve stampare un messaggio di errore e terminare.
- Nel caso tutte le condizioni siano soddisfatte, lo script deve stampare per ogni file tutte le righe incluse tra la riga di posizione `n1` e la riga di posizione `n2`. Se tali righe (del tutto o in part) non esistono (e.g., il file è più corto di `n1` o `n2` righe) lo script non deve visualizzare nulla per quello specifico file.

Ad esempio:

`filter.sh 5 7 foo bar`

deve visualizzare le righe `5`, `6` e `7` dei file `foo` e `bar`.

English

A script, whose name is `filter.sh`, receives on the command line two integers (`n1` and `n2`) and a set of strings representing file names, like:
`filter.sh n1 n2 file1 file2 file3 ...`
Write a script that:

- Checks that the number of command line parameters is greater than three, the fact that `n2` must be greater than `n1` (i.e., `n2>n1`), and the existence of all files in the filesystem.
- In case one of the previous conditions is false, the script must print an error message and terminate.
- In case all conditions are met, the script must display for each file all lines included from the line in position n1 and the line in position n2. If such lines (all or partially) do not exist (e.g., the file is shorter than n1 or n2 lines) the script should not display anything for that specific file.

For example
`filter.sh 5 7 foo bar`
displays lines `5`, `6`, and `7` of the `foo` and `bar` files.

Soluzione. Solution.

```
#!/bin/bash

# Checking argument number
if [ $# -lt 3 ] ; then
      echo "Usage: $0 n1 n2 file1 file2 file3 ..."
      exit 1
fi

# Reading inputs
n1=$1
n2=$2

let "diff=${n2}-${n1}+1"

# Checking n2 > n1
if [ ${n1} -ge ${n2} ] ; then
      echo "Error: n1 must be smaller than n2!"
      exit 2
fi

# Removing n1 and n2 from list of inputs
shift 2

# Checking file existence
for file in "$@" ; do
      if [ ! -f "${file}" ] ; then
            echo "Error: file '${file}' doesn't exist"
            exit 3
      fi
done

# Printing file contents
IFS='' # to preserve spaces with the 'while read' solution
for file in "$@" ; do
      # Checking file length
      if [ $(cat ${file} | wc -l) -le ${n2} ] ; then
            continue;
      fi

      # All of the following (and the while loop) are valid solutions
#     cat ${file} | head -n ${n2} | tail -n ${diff}
```

```
#       cat ${file} | tail -n +${n1} | head -n ${diff}
#       cat ${file} | cut -d $'\n' -f ${n1}-${n2}
#       sed -n "${n1},${n2}p" ${file}
#       awk "NR >= ${n1} && NR <= ${n2}" "${file}"

        index=0
        while read line ; do
                let "index++"
                if [ ${index} -ge ${n1} ] && [ ${index} -le ${n2} ] ; then
                        echo "${line}"
                fi
        done < ${file}
done
```

Ex 04 (3.0 points)

Italiano

Si consideri il seguente insieme di processi schedulati con un quantum temporale di 10 unità di tempo. Rappresentare, mediante diagramma di Gantt, l'esecuzione di tali processi utilizzando gli algoritmi First-Come First-Served (FCFS) e Priority Scheduling (PS). Si calcolino i tempi di terminazione di ciascun processo e il tempo di attesa medio. Si riportino i dati relativi all'algoritmo di scheduling con tempo medio di attesa MINORE. Si riporti la risposta su un'unica riga, indicando il nome dell'algoritmo (FCFS oppure PS), i tempi di terminazione di P1, P2, P3, P4 e P5 seguiti dal tempo di attesa medio. Separare numeri e stringhe con un unico spazio. Riportare il tempo di attesa medio con 1 sola cifra decimale. Non inserire nessun altro carattere nella risposta. Esempio di risposta corretta: PS 20 23 11 45 67 30.5

English

Consider the following processes, scheduled with a temporal quantum of 10 units. Represent, using a Gantt diagram, the execution of these processes using the First-Come First-Served (FCFS) and Priority Scheduling (PS) scheduling algorithms. Compute the termination time of each process and the average waiting time. Report the data only for the algorithm with the SMALLER average waiting time. Write your answer on a single line, indicating the name of the algorithm, the termination times of P1, P2, P3, P4, and P5 followed by the average waiting time. Separate the numbers with a single space. Report the average waiting time with a single decimal digit. Do not enter any other character in the response. Example of correct answer: PS 20 23 11 45 67 30.5

Processo Process	TempoArrivo ArrivalTime	BurstTime	Priorità Priority
P1	0	16	5
P2	3	23	4
P3	5	25	3
P4	7	13	2
P5	9	11	1

Fasi operative. Workflow.

```
FCFS:
  P1: from=  0 to  = 15 end = 16 waitingTime =  0 tournaroundTime = 16
  P2: from= 16 to  = 38 end = 39 waitingTime = 13 tournaroundTime = 36
  P3: from= 39 to  = 63 end = 64 waitingTime = 34 tournaroundTime = 59
  P4: from= 64 to  = 76 end = 77 waitingTime = 57 tournaroundTime = 70
  P5: from= 77 to  = 87 end = 88 waitingTime = 68 tournaroundTime = 79
  EndTime: 16 39 64 77 88
  AverageWaitingTime: (  0 + 13 + 34 + 57 + 68 ) / 5 = 34.40
  AverageTournaroudTime: ( 16 + 36 + 59 + 70 + 79 ) / 5 = 104.00
GanttChart:
1111111111111112222222222222222222222333333333333333333333333444444444444455555555
5555
```

```
PS:
  P1: from=  0 to  = 15 end = 16 waitingTime =  0 tournaroundTime = 16
  P5: from= 16 to  = 26 end = 27 waitingTime =  7 tournaroundTime = 24
  P4: from= 27 to  = 39 end = 40 waitingTime = 20 tournaroundTime = 35
  P3: from= 40 to  = 64 end = 65 waitingTime = 35 tournaroundTime = 58
  P2: from= 65 to  = 87 end = 88 waitingTime = 62 tournaroundTime = 79
  EndTime: 16 27 40 65 88
  AverageWaitingTime: (  0 +  7 + 20 + 35 + 62 ) / 5 = 24.80
  AverageTournaroudTime: ( 16 + 24 + 35 + 58 + 79 ) / 5 = 84.80
GanttChart:
1111111111111111555555555554444444444444333333333333333333333333222222222222222222222222
```

Risposta. Answer.

PS 16 88 65 40 27 24.8

Ex 05 (3.0 points)

Italiano

Lo stato di un sistema con 5 processi e 3 tipologie di risorse è definito dalla matrice riportata alla fine della domanda. Si osservi che, a differenza del solito, la seconda colonna della tabella NON riporta le risorse già assegnate BENSI' quelle ancora necessarie (cioè ancora richiedibili dal processo nell'ambito delle richieste massime).

Si supponga il processo P4 effettui una richiesta per le risorse {1, 1, 1}. Si stabilisca se la richiesta può essere soddisfatta e permetta al sistema di rimanere in uno stato sicuro. In caso affermativo si risponda "YES" e si riporti la sequenza di esecuzione dei processi. Ad esempio, si risponda "YES 5 4 3 2 1" nel caso in cui la sequenza sicura sia P5, P4, P3, P2, P1. In caso contrario, si risponda "NO".

English

The state of a system with 5 processes and 3 types of resources is defined by the matrices reported at the end of the question. Note that, unlike usual, the second column of the table does NOT show the resources already allocated BUT those still needed (i.e., still requestable by the process among the maximum possible resources that can be requested).

Suppose that process P4 requests the resources {1, 1, 1}. Indicate whether or not the request can be granted, allowing the system to remain in a safe state. In the affirmative case, respond with "YES" and report the sequence in which the processes will be executed. For instance, respond "YES 5 4 3 2 1" in the case the safe sequence is P5, P4, P3, P2, P1. Otherwise, respond with "NO".

Processo **Process**	**Necessità** **Need**	**Massimo** **Max**	**Disponibile** **Available**
P1	4 3 5	4 3 5	3 2 3
P2	3 2 2	3 4 2	
P3	1 5 3	2 5 4	
P4	3 1 2	3 2 2	
P5	4 4 1	4 4 2	

Fasi operative. Workflow.

Step 0; Disponibili: 2 1 2
Step: 1 RunProc: 4 Necessita': 2 0 1 --> Disponibili: 3 3 3
Step: 2 RunProc: 2 Necessita': 3 2 2 --> Disponibili: 3 5 3
Step: 3 RunProc: 3 Necessita': 1 5 3 --> Disponibili: 4 5 4
Step: 4 RunProc: 5 Necessita': 4 4 1 --> Disponibili: 4 5 5
Step: 5 RunProc: 1 Necessita': 4 3 5 --> Disponibili: 4 5 5

Risposta. Answer.
YES 4 2 3 5 1

Ex 06 (2.5 points)

Italiano
Il direttorio di lavoro corrente (cioè, .) contiene i file riportati di seguito, il cui elenco è stato derivato mediante il comando ls -l.

```
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:18 aaXXXb.txt
-rwxrwxr-x 1 scanzio scanzio    0 gen  2 22:17 aXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:17 aXXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:17 aXXXXXb.sh
-r--r--r-- 1 scanzio scanzio    0 gen  2 22:17 aXXXXXb.txt
drwxrwxr-x 2 scanzio scanzio 4096 gen  2 22:22 aXXXXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:23 aXXXXXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:22 aXXXXXXXXXb.txt
```

Quali sono i file selezionati dal comando find che segue:

```
find . -type f -regextype posix-extended -regex "\.\/aXXX(XX)*b\.txt" -writable
```

Si osservi che risposte errate implicano una penalità nel punteggio finale.

English
The current working directory (i.e., .) contains the following files, which were derived from the output of the command ls -l.

```
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:18 aaXXXb.txt
-rwxrwxr-x 1 scanzio scanzio    0 gen  2 22:17 aXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:17 aXXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:17 aXXXXXb.sh
-r--r--r-- 1 scanzio scanzio    0 gen  2 22:17 aXXXXXb.txt
drwxrwxr-x 2 scanzio scanzio 4096 gen  2 22:22 aXXXXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:23 aXXXXXXXb.txt
-rw-rw-r-- 1 scanzio scanzio    0 gen  2 22:22 aXXXXXXXXXb.txt
```

What are the files selected by the find command that follows:

```
find . -type f -regextype posix-extended -regex "\.\/aXXX(XX)*b\.txt" -writable
```

Note that wrong answers imply a penalty in the final score.

Scegli una o più alternative. Choose one or more options.

1. ⬜ `./aaXXXb.txt`
2. ✅ `./aXXXb.txt`
3. ⬜ `./aXXXXb.txt`
4. ⬜ `./aXXXXXb.sh`
5. ⬜ `./aXXXXXb.txt`
6. ⬜ `./aXXXXXXb.txt`
7. ✅ `./aXXXXXXXb.txt`
8. ✅ `./aXXXXXXXXXb.txt`

Ex 07 (2.0 points)

Italiano
In riferimento ai segnali, si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English
Referring to signals, indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Scegli una o più alternative. Choose one or more options.

1. ⬜ Un segnale inviato dalla system call `kill()` può essere mascherato; un segnale inviato dal comando di shell `kill` non può esserlo. A signal sent by the system call `kill()` can be masked; a signal sent by the `kill` shell command cannot.
2. ✅ Sia il comando di shell `kill` che la system call `kill()`, possono uccidere un processo. Both the shell command `kill` and the system call `kill()` can kill a process.
3. ✅ Sia la system call `kill()` che il comando di shell `kill` possono essere utilizzate per inviare un segnale a un processo. Both the system call `kill()` and the shell command `kill` may be used to send a signal to a process.
4. ⬜ Attraverso la system call `signal()` si può decidere di ignorare QUALSIASI tipo di segnale. By using the system call `signal()` you can decide to ignore ANY type of signal.
5. ✅ La ricezione di alcuni tipi di segnali (ad esempio SIGCHLD) non può essere ignorata. The reception of some types of signal (for instance SIGCHLD) cannot be ignored.
6. ⬜ All'interno di un signal handler possono essere usate anche funzioni non rientranti senza nessuna ripercussione sulla correttezza del programma. Inside a signal handler non-reentrant functions can also be used without any change in the correctness of the program.
7. ⬜ La system call `kill()` uccide un processo, il comando di shell `kill` non lo uccide. The system call `kill()` kills a process, the shell command `kill` does not kill it.
8. ✅ L'esecuzione di un signal handler a seguito della ricezione di un segnale da parte di un processo può portare ad avere delle race condition. The execution of a signal handler after the reception of a signal by a process can lead to have race conditions.

Ex 08 (2.0 points)

Italiano

Si considerino i metodi di codifica delle informazioni su file. Si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Consider the methods for encoding information in a file. Indicates which of the following observations are correct (possibly more than one). Note that incorrect answers may imply a penalty on the final score.

Scegli una o più alternative. Choose one or more options.

1. ⬜ I caratteri in UTF-8 sono sempre salvati su 32 bits. UTF-8 characters are always stored on 32 bits.
2. ✅ UTF-8 è un'estensione dell'extended ASCII usata per rappresentare almeno 256 caratteri. UTF-8 is an extension of ASCII used to represent at least 256 characters.
3. ✅ I caratteri in UTF-8 possono essere codificati su 16 bits. UTF-8 characters can be stored on 16 bits.
4. ⬜ Un numero (e.g., intero o reale) codificato in ASCII o in binario ha la medesima rappresentazione in termini di sequenza di bit. A number (e.g., integer or float) encoded in ASCII or binary has the same representation in terms of sequence of bits.
5. ⬜ La codifica di un numero intero in ASCII è sempre più compatta della codifica dello stesso numero in intero binario. Encoding an integer number in ASCII is always more compact than encoding the same number in integer binary.
6. ✅ Lo stesso file binario potrebbe essere interpretato in modo differente se letto da due calcolatori differenti. The same binary file could be interpreted differently when read by two different computers.
7. ✅ I file ASCII sono spesso più compatti rispetto ai file codificati in UTF-8. ASCII files are often more compact than UTF-8 files.

Ex 09 (2.5 points)

Italiano

Il seguente script, di nome `script.sh`:

```
echo $#
shift
echo $*
exit 0
```

viene eseguito con la seguente riga di comando:
`a=3; source script.sh one two "${a}2" sss`
Si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.
English
The following script, named `script.sh`:

```
echo $#
shift
echo $*
exit 0
```

is run with the following command line:
`a=3; source script.sh one two "${a}2" sss`
Indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

Scegli una o più alternative. Choose one or more options.

1. ✅ Esiste almeno una riga che contiene il seguente output. There is at least one line that contains the following output:
   `4`
2. ⬜ Esiste almeno una riga che contiene il seguente output. There is at least one line that contains the following output:
   `one two 32`
3. ✅ Esiste almeno una riga che contiene il seguente output. There is at least one line that contains the following output:
   `two 32 sss`
4. ⬜ Esiste almeno una riga che contiene il seguente output. There is at least one line that contains the following output:
   `5`
5. ✅ Alla fine dell'esecuzione il terminale che ha lanciato lo script viene chiuso. At the end of execution, the terminal that launched the script is closed.
6. ⬜ Esiste almeno una riga che contiene il seguente output. There is at least one line which contains the following output.
   `one two a2`
7. ⬜ Esiste almeno una riga che contiene il seguente output. There is at least one line which contains the following output.
   `two a2 sss`
8. ✅ Il codice contenuto in `script.sh` produce dell'output anche se il file script.sh non ha permesso di esecuzione. The code contained in `script.sh` produces its output even if the script.sh file does not have the execution permission.

Ex 10 (2.0 points)

Italiano
Si analizzi il seguente tratto di codice. Si indichi quanti caratteri 'X' vengono visualizzati.
English
Analyze the following segment of code. Indicate how many characters 'X' are displayed.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main () {
  int x;
  x = 1;
  setbuf(stdout,0);
```

```
  while (x<=4 && fork()) {
    if (!fork ()) {
      if (fork())
        exit(0);
      printf ("X");
    }
    x++;
  }
  return 1;
}
```

Scegli UNA SOLA alternativa: Choose JUST ONE option:

1. ⬜ 20
2. ⬜ 30
3. ⬜ 40
4. ✅ 15
5. ⬜ 7

Ex 11 (2.5 points)

Italiano

Dato il seguente Makefile, si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Given the following Makefile, indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

```
.PHONY: A B C D E F G H I

E:
	@echo -n "E"
B: E F
	@echo -n "B"
C: G H
	@echo -n "C"
G: I
	@echo -n "G"
F:
	@echo -n "F"
H:
	@echo -n "H"
I:
	@echo -n "I"
D:
	@echo -n "D"
A: B C D
	@echo -n "A"
CMD:
	@echo "pippo" > file
	@cat file | tr -s p | tr p v
```

Scegli una o più alternative. Choose one or more options.

1. ✅ Il comando `make` produce il seguente output. The command `make` produces the following output.
   ```
   E
   ```

2. ⬜ Il comando `make G` produce il seguente output. The command `make G` produces the following output.
   `G`
3. ⬜ Il comando `make B` produce il seguente output. The command `make B` produces the following output.
   `BEF`
4. ✅ Il comando `make A` produce il seguente output. The command `make A` produces the following output.
   `EFBIGHCDA`
5. ⬜ Se il comando `make F` viene eseguito due volte, la seconda volta non produce nessun output. If the command `make F` is executed twice, the second time it does not produce any output.
6. ✅ Il comando `make CMD` produce il seguente output. The command `make CMD` produces the following output.
   `vivo`

Ex 12 (3.0 points)

Italiano

Si indichi l'output o gli output possibili che possono essere ottenuti eseguendo concorrentemente i processi PA, PB e PC riportati di seguito. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Indicate the possible output or outputs that we can obtain by concurrently executing the following processes PA, PB, and PC. Note that wrong answers imply a penalty in the final score.

```
init (S1, 1); init(S2, 0); init(S3, 0);
```

PA
```
printf("A");
signal(S3);
wait(S1);
printf("B");
wait(S1);
printf("C");
signal(S2);
```

PB
```
wait(S3);
printf("D");
signal(S1);
wait(S2);
printf("E");
```

PC
```
wait(S2);
printf("F");
```

Scegli una o più alternative. Choose one or more options.

1. ✅ ADBCE
2. ⬜ ABDCEF
3. ✅ ABDCF
4. ✅ ADBCF
5. ⬜ ADBCEF
6. ✅ ABDCE
7. ⬜ ABCDE
8. ⬜ ADEBC
9. ⬜ ADBEC