Esame Sistemi Operativi - Operating Systems Exam
2023/06/15

Ex 1 (4.0 points)

Italiano

Si supponga che il seguente programma venga eseguito mediante l'istruzione `./pgrm 2`. Si indichi un possibile output del programma.

English

Suppose that the following program is run using the command `./pgrm 2`. Indicate a possible output of the program.

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

int main (int argc, char *argv[]) {
  int i, n;
  char str[40];

  n = atoi (argv[1]);

  for (i=1; i<=n; i++) {
    if ( fork() == 0) {
      sprintf (str, "%d", n-1);
      execlp (argv[0], argv[0], str, NULL);
    }
  }

  printf ("n=%d\n", n); fflush (stdout);

  exit (0);
}
```

Risposta. Answer.

```
n=2
n=1
n=1
n=0
n=0
```

Ex 2 (2.0 points)

Italiano

Data la seguente riga di comando bash:

```
egrep -e "^H" in.txt | egrep -e "H[23][0-9]" | egrep -v "\-X$"| cut -d "-" -f1,3
```

Riportare quali righe sono stampate in output dal precedente comando, quando esso è eseguito sul file input.txt riportato alla fine della domanda.

English

Given the following bash command:

```
egrep -e "^H" in.txt | egrep -e "H[23][0-9]" | egrep -v "\-X$"| cut -d "-" -f1,3
```

Report which lines are printed in output from the previous command when it is executed on the file input.txt, which is reported at the end of the question.

Contenuto del file `in.txt` Content of the file `in.txt`

```
H21-5.60-Stefano-XX
B23-2.34-Stefano-X
B22-4.50-Giulia-XX
H25-2.20-Gabriele-XX
H10-8.40-Apinda-X
H30-2.30-Gabriele-X
H30-2.40-Giulia-XX
```

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

1. ✅ `H30-Giulia`
2. ⬜ `H21-5.60-Stefano`
3. ⬜ `H30-2.40-Giulia`
4. ✅ `H21-Stefano`
5. ⬜ `H25-2.20-Gabriele`
6. ⬜ `B22-Giulia`
7. ✅ `H25-Gabriele`
8. ⬜ `B22-4.50-Giulia`
9. ⬜ `H30-2.30-Gabriele`
10. ⬜ `H30-Gabriele`

Ex 3 (3.0 points)

Italiano

Sia dato il seguente makefile eseguito mediante l'istruzione `make` (senza nessun parametro). Si indichi un possibile output del programma.

English

Suppose that the following makefile is run using the command `make` (without any parameters). Indicate a possible output of the program.

```
.PHONY: t1 t2 t3 t4 t5 t6 t7 t8 t9 t10

t2: t5 t6
	echo "t2"
t3: t7 t8
	echo "t3"
t6: t9 t10
	echo "t6"
t4:
	echo "t4"
t1: t2 t3 t4
	echo "t1"
t5:
	echo "t5"
t7:
	echo "t7"
t8:
	echo "t8"
t9:
	echo "t9"
t10:
	echo "t10"
```

Soluzione. Solution.

```
echo "t5"
t5
echo "t9"
t9
echo "t10"
t10
echo "t6"
t6
echo "t2"
t2
```

O: Or:

```
t5
t9
t10
t6
t2
```

Ex 4 (2.0 points)

Italiano

Si supponga un processo diventi "zombie".
Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose that a process becomes "zombie".
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. ✅ Il PCB del processo è ancora presente in memoria The PCB of the process is still present in the memory
2. ⬜ Il processo zombie ha effettuato una wait The zombie process performed a wait
3. ⬜ Il padre del processo zombie ha effettuato una wait The parent of the zombie process performed a wait
4. ✅ Il padre del processo zombie non ha ancora effettuato una wait (o una waitpid) The parent of the zombie process has not made a wait (or a waitpid) yet
5. ⬜ Il processo non ha processo padre The process does not have a parent process
6. ⬜ Il PCB del processo padre verrà cancellato quando il processo figlio effettua una wait The PCB of the parent process will be deleted only after the child process performs a wait
7. ⬜ Il processo è stato sicuramente ereditato dal processo "init" The process was certainly inherited by the process “init”
8. ✅ Il suo PCB verrà cancellato solo dopo che il padre effettuerà una wait o una waitpid Its PCB will be deleted only after its parent performs a wait or a waitpid

Ex 5 (2.0 points)

Italiano

Si faccia riferimento all scheduling dei processi.
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Consider the scheduling of processes.
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. ⬜ La strategia shortest job first (SJF) usa la prelazione per la schedulazione dei processi The strategy shortest job first (SJF) uses preemption for processes scheduling
2. ✅ Nel caso di algoritmi di schedulazione con prelazione può avvenire l'operazione di context switching In the case of scheduling algorithms with preemption, the context switching operation can take place
3. ⬜ Un processo inserito nella coda "waiting" può passare direttamente nello stato "running" A process inserted in the "waiting" queue can pass directly in the "running" state
4. ✅ Il "turnaround time" è il tempo che trascorre dalla sottomissione alla terminazione di un processo The "turnaround time" is the time from the submission to the termination of a process
5. ⬜ Il "waiting time" è la somma del tempo trascorso da un processo nella coda "waiting" The "waiting time" is the sum of the time spent by the process in the "waiting" queue
6. ✅ La strategia shortest job first (SJF) è ottima dal punto di vista del "waiting time" The strategy shortest job first (SJF) is optimal from the point of view of the "waiting time"

Ex 6 (3.0 points)

Italiano

Dati tre processi PA, PB, PC e PD il cui codice è riportato a seguire e i cui pid siano pid_PA, pid_PB, pid_PC e pid_PD, rispettivamente. Si indichino quali dei seguenti output sono corretti. Si assuma che la funzione `other_code()` non contenga al suo interno chiamate bloccanti o altre chiamate alla system call `kill()`. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Given three processes PA, PB, PC e PD whose code is reported in the following and whose pids are pid_PA, pid_PB, pid_PC and pid_PD, respectively. Indicate which of the following outputs are correct. Assume that the `other_code()` function contains neither calls to other blocking functions nor calls to the `kill()` system call. Note that incorrect answers imply a penalty in the final score.

```
PA
kill(pid_PB, SIG…);
pause();
printf("A");
PB
other_code();
pause();
printf("B");
kill(pid_PC, SIG…);
kill(pid_PD, SIG…);
PC
other_code();
pause();
printf("C");
PD
other_code();
pause();
printf("D");
```

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

1. ✅ Nessun output. No output.
2. ✅ B
3. ✅ BC
4. ✅ BD
5. ✅ BCD
6. ⬜ BACD
7. ⬜ BADC

Ex 7 (3.0 points)

Italiano

Si indichi l'output o gli output possibili che possono essere ottenuti eseguendo concorrentemente i processi PA, PB e PC riportato di seguito.

English

Indicate the possible output or outputs that can be obtained by concurrently executing the processes PA, PB e PC as follow

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

PA

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

PB

```
wait(S2);
printf("D");
signal(S1);
```

PC

```
wait(S2);
printf("E");
signal(S1);
```

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

1. ⬜ DABC
2. ⬜ DABCE
3. ⬜ EABCD
4. ⬜ EABC
5. ✅ DAE
6. ⬜ DAEBC
7. ✅ EAD
8. ⬜ EADBC

Ex 8 (3.0 points)

Italiano

Con riferimento ai kernel-level e agli user-level thread si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

With reference to kernel-level and user-level threads, 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. ✅ Nel caso di kernel-level thread il sistema operativo mantiene una tabella globale di informazioni sui thread. In the case of kernel-level threads, the operating system maintains a global table of thread information.
2. ✅ Ogni kernel-thread può essere utilizzato a turno da uno user-thread. Each kernel thread can be used in turn by a user thread.
3. ✅ Gli user-thread possono essere molto più numerosi dei kernel-thread. User threads can be much more numerous than kernel threads.
4. ⬜ Nel caso di user-level thread il sistema operativo mantiene una tabella globale di informazioni sui thread. In the case of user-level threads, the operating system maintains a global table of thread information.
5. ⬜ I kernel-level thread sono normalmente più efficienti degli user-level thread. Kernel-level threads usually are more efficient than user-level threads.
6. ⬜ Normalmente nei sistemi operativi per ogni user-level thread esiste un kernel-level thread. Typically in operating systems, there is a kernel-level thread for each user-level thread.

7. ⬜ Ogni user-thread può essere utilizzato a turno da un kernel-thread. In turn, each user thread can be used by a kernel thread.

Ex 9 (4.0 points)

Italiano

Realizzare uno script BASH che permetta di riconoscere e risolvere le coppie di direttive `#ifdef-#endif` all'interno di un file C. Lo script riceve sulla riga di comando il nome di due file. Verificare il corretto passaggio dei parametri. Nel primo file lo script deve riconoscere tutti i blocchi di righe racchiuse dalle coppie di direttive

```
#ifdef 1
…
#endif
```

oppure

```
#ifdef 0
…
#endif
```

Lo script deve creare una copia del file C originale, usando come nome quello specificato dal secondo parametro, risolvendo tutte le coppie di direttive `#ifdef-#endif` riconosciute. Per risolvere una coppia di direttive `#ifdef-#endif`, lo script deve eliminare la coppia di direttive stesse e tutte le tutte le righe racchiuse tra di esse nel caso in cui la condizione di `#ifdef` sia `0` oppure eliminare solo la coppia di direttive mantenendo invece le righe racchiuse tra di esse nel caso in cui cui la condizione di `#ifdef` sia `1`.

English

Create a BASH script to recognize and solve all pairs of `#ifdef-#endif` directives in a C file. The script receives the names of two files on the command line. Check that the correct parameters are passed to the script. In the first file, the script should recognize all block of lines that are surrounded by the pairs of directives:

```
#ifdef 1
...
#endif
```

or

```
#ifdef 0
...
#endif
```

The script should create a copy of the original file, using the name passed as the second command line parameter, solving all pairs of `#ifdef-#endif` directives found. To solve a pair of `#ifdef-#endif` directives, the script should delete the directives along with all the lines between them in case the condition of `#ifdef` is `0` or delete directives only while keeping all the lines between them in case the condition of `#ifdef` is `1`.

Soluzione. Solution.

```
#!/bin/bash
################################################################################
# Version 1: nested loop and if statements
################################################################################

# Check if the correct parameters has been passed
if [ $# -lt 2 ]; then
  echo "Usage: ./ifdef.sh <input_file> <output_file>"
  exit 1
fi

# Check if the first parameter is a valid filename
if [ ! -f $1 ]; then
      echo "The first parameter should be the path to a valid file"
      exit 1
fi

# Process the input file line-by-line
```

```
while read line; do

      # Check if a conditional block starts at the current line
      if [ "$(echo $line | cut -d " " -f 1)" == "#ifdef" ]; then

      # Initialize flag to keep or discard lines in the conditional block
      keep=$(echo $line | cut -d " " -f 2)

      # Read the conditional block
      while read line; do

            # Check if a conditional block ends at the current line
            if [ "$(echo $line | cut -d " " -f 1)" == "#endif" ]; then
            break
            elif [ $keep == "1" ]; then
            echo $line >> $2
            fi
      done
      else
      # Echo all lines non in a conditional block
      echo $line >> $2
      fi

done < $1


#!/bin/bash
#################################################################################
# Version 2: state variable
#################################################################################

# Check if the correct parameters has been passed
if [ $# -lt 2 ]; then
  echo "Usage: ./ifdef.sh <input_file> <output_file>"
  exit 1
fi

# Check if the first parameter is a valid filename
if [ ! -f $1 ]; then
      echo "The first parameter should be the path to a valid file"
      exit 1
fi

# Process the input file line-by-line
block_found=0
while read line; do

      # If we are outside a conditional block either detect the start of a new block
or echo the current line
      if [ $block_found -eq 0 ]; then
      if [ "$(echo $line | cut -d " " -f 1)" == "#ifdef" ]; then
            block_found=1
            keep=$(echo $line | cut -d " " -f 2)
      else
            echo $line >> $2
      fi
```

```
      # If we are inside a conditional block either detect the end of the block or
decide wether to echo the line or not
      elif [ $block_found -eq 1 ]; then
      if [ "$(echo $line | cut -d " " -f 1)" == "#endif" ]; then
            block_found=0
      elif [ $keep -eq 1 ]; then
            echo $line >> $2
      fi
      fi

done < $1
```

Ex 10 (4.0 points)

Italiano

Si descriva il problema dei Readers e Writers nel caso di precedenza ai Writers. Si indichi esattamente che cosa significa dare precedenza ai Writers; si indichi inoltre il significato delle varie primitive di sincronizzazione utilizzate.

English

Describe the problem of the Readers and Writers in the case of precedence to Writers. Indicate exactly what it means to give precedence to Writers; moreover, indicate the meaning of the various synchronization primitives used.

Soluzione. Solution.

It is a classical problem about shared data between two sets of concurrent processes: a set of Readers, which can access concurrently to the data; a set of Writers, which can access in mutual exclusion, both with other Writers and Readers processes, to the data.
Giving priority to writers means a ready Writer, must wait the shortest possible time.

```
nR=nW=0; // Count the number of readers and writers within the critical section
init (w, 1); // Blocks the first reader when writers are in the critical section
init (r, 1); // Blocks the first writer when readers are in the critical section,
             // and its presence in the reader code is used to provide precedence
             // to writers
init (meR, 1); // Used for mutual exclusion (variable nR)
init (meW, 1); // Used for mutual exclusion (variable nW) and to block starting
               // from the second writer when readers are in the critical section

Reader:
wait(r);
wait (meR);
  nR++;
  if (nR == 1)
    wait (w);
signal (meR);
signal (r);
... reading ...
wait (meR);
  nR--;
  if (nR == 0)
    signal (w);
signal (meR);

Writer:
wait (meW);
  nW++;
```

```
  if (nW == 1)
    wait (r);
signal (meW);
wait (w);
... writing ...
signal (w)
wait (meW);
  nW--;
  if (nW == 0)
    signal (r);
signal (meW);
```

Ex 11 (6.0 points)

Italiano

Si scriva un programma C in grado di eseguire tre thread denominati TA, TB e TC. Tali thread sono ciclici e per ciascuna iterazione del proprio ciclo visualizzano un carattere 'A', 'B', oppure 'C', rispettivamente. Sincronizzare i tre thread all'interno dei costrutti iterativi in modo che possano essere visualizzate solo due sequenze di caratteri, ovvero `ABC` oppure `CBA`. Dato che non ci si deve occupare della terminazione dei thread, la seguente sequenza di stampa generata dallo loro esecuzione è corretta:
`ABCABCABCCBAABCCBACBACBA…`

English

Write a C program that executes three threads named TA, TB, and TC. These threads are cyclical and display a character 'A', 'B', or 'C' for each iteration of their loop, respectively. Synchronize the three threads within the iterative construct so that only two sequences of characters can be displayed, namely `ABC` or `CBA`. Since the thread termination does not have to be addressed, the following sequence generated by the execution of TA, TB, and TC is correct:
`ABCABCABCCBAABCCBACBACBA...`

Soluzione. Solution.

```
sem_t s1, s2, s3;
int first=1;

sem_init(&s1,0,1);
sem_init(&s2,0,0);
sem_init(&b,0,0);


TA:
while(1){
  sem_wait(&s1);
  printf("A");
  if (first){
    first=0;
    sem_post(&s2);
    sem_wait(&b);
    sem_post(&s1);
  }else{
    first=1;
    sem_signal(&b);
  }
}

TB:
while(1){
  sem_wait(&s2);
  printf("B");
```

```
  sem_post(&s1);
}

TC:
while(1){
  sem_wait(&s1);
  printf("C");
  if (first){
    first=0;
    sem_post(&s2);
    sem_wait(&b);
    sem_post(&s1);
  }else{
    first=1;
    sem_signal(&b);
  }
}
```