Processi / Processes

Exam 2021/01/29 - Ex 2 (3.0 points)

Italiano

Si supponga di eseguire il seguente programma. Si riporti l'output da esso generato rispettando esattamente il formato prodotto.
Si supponga che il programma venga eseguito su un sistema operativo nel quale l'attesa di 1 secondo sia sufficientemente lunga per completare tutti gli altri task in esecuzione in quel momento.
Si prega di riportare la risposta su un'unica riga, indicando i vari messaggi/valori in output separati da un unico spazio. Non inserire nessun altro carattere nella risposta.

English

Suppose you are running the following program. Report the output it generates, remembering to respect exactly the output format produced.
Assume that the program is running on an operating system where a sleep of 1 second is long enough to complete all the other tasks currently running.
Please, write the answer on a single line, indicating the various output messages/values separated by a single space. Do not enter any other character in the response.

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

int main(){
	int x;
	x=0;
	while (x<2 && fork()){
		if (!fork())
			execlp ("echo", "x++", "x", NULL);
		x++;
		sleep (1);
		system("echo x+x");
	}
}
```

Risposta: Answer:

```
x x+x x x+x
```

Exam 2021/02/12 - Ex 4 (1.5 points)

Italiano

Si supponga un processo diventi "orfano".
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose a process becomes an "orphan."
Please 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. ⬜ Il processo sta attendendo che il padre effettui una wait The process is waiting that the parent performs a wait

2. ⬜ Il processo è diventato orfano in quanto non ha effettuato una wait The process becomes an orphan because it did not perform a wait
3. ⬜ Il processo diventerà "zombie" alla sua terminazione The process will become "zombie" at its termination
4. ✅ Il processo viene ereditato dal processo "init" The process is inherited by the "init" process
5. ✅ Il processo non diventerà "zombie" alla sua terminazione perché il processo "init" lo erediterà The process will not become "zombie" at its termination because the "init" process will inherit it

Exam 2021/02/12 - Ex 6 (1.5 points)

Italiano

Si analizzi il seguente tratto di codice.
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale

English

Analyze the following piece of code.
Please indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

```
if (fork() == 0) {
      /* first child */
      if (fork() == 0) {
            /* second child */
            ...
      } else {
            exit (1);
      }
}
wait ();
```

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

1. ⬜ Il padre può soffrire di deadlock The parent can suffer of deadlock
2. ✅ Nel caso in cui il primo figlio abbia eseguito la exit(1), il secondo figlio non diventerà mai zombie In case the first child has performed exit(1) , the second child will never become a zombie
3. ⬜ Affinchè il codice sia corretto occorre inserire una seconda wait al termine To have a correct piece of code we must insert a second wait statement at the end
4. ✅ Quando il primo figlio termina, il secondo figlio viene ereditato da “init” When the first child terminates, the second child is inherited by “init”
5. ✅ Il padre attende la terminazione del primo figlio The parent waits the termination of the first child
6. ⬜ Il padre attende la terminazione del primo e del secondo figlio The parent waits the termination of the first and the second child

Exam 2021/01/29 - Ex 1 (3.0 points)

Italiano

Un programma concorrente è costituito da un unico processo di nome P1 (e funzione P1()), di tipo ciclico, di cui sono presenti 2 istanze.
Il comportamento del programma è il seguente:

- All’inizio le 2 istanze del processo P1 sono eseguite in parallelo.
- Quando entrambe le istanze di P1 hanno finito di eseguire il proprio codice (quello presente nella funzione P1()), entrambe le istanze di P1 sono eseguite nuovamente.

Si indichino quali dei seguenti codici sono corretti. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

A concurrent program is composed of a single process called P1 (and a function P1()). The process, and consequently the function, are cyclic. There are 2 instances of process P1.
The behavior of the program is as follows:

- At the beginning, the 2 instances of process P1 are executed in parallel.
- When both instances of P1 have finished executing their code (i.e., the code that is present in function P1()), both instances of P1 are executed again.

Indicate which of the following codes are correct. Note that wrong answers imply a penalty in the final score

Scegli UNA SOLA alternativa: Choose JUST ONE option:

1. ⬜
```
int n=0;
init(s, 2);
init(m, 1);
init(b, 0);
while(1) {
	wait(s);
	P1();
	wait(m);
	n++;
	if (n==2) {
		signal(s);
		signal(s);
		n=0;
		signal(b);
	} else {
		signal(m);
		wait(b);
	}
}
```
2. ✅
```
int n=0;
init(s, 2);
init(m, 1);
init(b, 0);
while(1) {
	wait(s);
	P1();
	wait(m);
	n++;
	if (n==2) {
		signal(s);
		signal(s);
		n=0;
		signal(b);
		signal(m);
	} else {
		signal(m);
		wait(b);
	}
}
```
3. ⬜
```
int n=0;
init(s, 2);
init(m, 1);
while(1) {
	wait(s);
	P1();
	wait(m);
```

```
                n++;
                if (n==2) {
                        signal(s);
                        signal(s);
                        n=0;
                }
                signal(m);
        }
```

Exam 2020/06/16 - Ex 2 (6.0 points)

Italiano

Si illustrino le caratteristiche dei segnali per inviare informazioni asincrone tra processi. Si riporti un programma che ne illustri l'utilizzo effettuando le seguenti operazioni:

1. Il programma principale crea due processi figlio, P1 e P2 e rimane in attesa di ricevere segnali da tali processi. Ogni volta che riceve un segnale da P1 visualizza su standard output il messaggio "Segnale ricevuto da P1". Analogamente, ogni volta che riceve un segnale da P2 visualizza su standard output il messaggio "Segnale ricevuto da P2". Se però riceve 3 segnali consecutivi dallo stesso processo, uccide i processi P1 e P2, utilizzando il comando di shell "kill" e termina.
2. I processi P1 e P2 eseguono un ciclo infinito, all'interno del quale attendono un tempo casuale e poi inviano un messaggio al processo padre. P1 trasferisce segnali di tipo SIGUSR1; P2 trasferisce segnali di tipo SIGUSR2.

English

Describe the characteristics of the signals to send asynchronous information between processes. Implement the following program:

1. A program generates two processes P1 and P2 and it awaits for receiving signals from them. Every time it receives a message from P1 in displays on standard output the message "Signal received from P1". Analogously, every time it receives a message from process P2 it displays the message "Signal received from P2". If it receives 3 signals from the same process, it terminates processes P1 and P2 using the shell command "kill" and it terminates.
2. Process P1 and P2 run through an infinite cycle. Within the cycle, they await for a random time and then send a signal to the parent. P1 sends signal SIGUSR1; P2 sends signals SIGUSR2.

Risposta: Answer:

```
/* Exam 2020/06/16 - Exercise 2
      Describe the characteristics of the signals to send asynchronous information
between processes: See slides. */
#include<signal.h>
#include<sys/types.h>
#include<sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int last_sig = -1;
int last_last_sig = -1;
int finish = 0;

void sign_handler(int sig){
      if (sig==SIGUSR1)
            printf("Signal received from P1\n");
      else if (sig==SIGUSR2)
            printf("Signal received from P2\n");
```

```
	/* It receives 3 consecutive signals from the same process */
	if (sig == last_sig && last_sig == last_last_sig) {
		finish = 1;
	} else {
		last_last_sig = last_sig;
		last_sig = sig;
	}
}

int main() {
	char cmd[100];
	pid_t pid1, pid2;
	if ( (signal(SIGUSR1, sign_handler) == SIG_ERR) || (signal(SIGUSR2,
sign_handler ) == SIG_ERR) ) {
		printf("Error initializing signal handler");
		exit(-1);
	}
	pid1 = fork();
	if (!pid1) {
		/* P1 */
		while (1) {
			sleep( rand()%2 );
			kill(getppid(), SIGUSR1);
		}
	} else {
		pid2 = fork();
		if (!pid2) {
			/* P2 */
			while (1) {
				sleep( rand()%3 );
				kill(getppid(), SIGUSR2);
			}
		}
	}
	/* Parent */
	while (1) {
		pause();
		if (finish) {
			/* Kill P1 with a shell command */
			sprintf(cmd, "kill -9 %d", pid1);
			system(cmd);
			/* Kill P2 with a shell command */
			sprintf(cmd, "kill -9 %d", pid2);
			system(cmd);
			exit(0);
		}
	}
}
```

Exam 2021/06/18 - Ex 11 (1.5 points)

Italiano

Si supponga un processo effettui una "waitpid". Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale

English

Suppose a process does a waitpid. Please 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. ✅ Il processo riceverà un SIGCHLD non appena un suo figlio termina. The process will receive a SIGCHLD as soon as one of its children ends
2. ⬜ Il processo potrà attendere un massimo numero di secondi. The process can wait a maximum number of seconds
3. ⬜ Il processo uscirà dalla waitpid alla terminazione del suo primo figlio. The process will exit from waitpid at the end of its first child
4. ✅ Il processo potrà rimanere bloccato sulla waitpid anche dopo la terminazione di un figlio. The process may get stuck on the waitpid even after a child has terminated

Exam 2021/01/29 Ex 9 (1.5 points)

Italiano

Relativamente all'implementazione delle pipe in un sistema operativo Unix, si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Regarding the implementation of pipes in a Unix operating system, 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. ✅ La scrittura su una pipe in cui tutti gli estremi di lettura sono stati chiusi provoca la generazione di un segnale di tipo SIGPIPE The write operation on a pipe, in which all reading extremes have been closed, causes the generation of a SIGPIPE signal
2. ⬜ Il tentativo di scrittura su una pipe piena restituisce errore The write attempt on a full pipe returns an error
3. ✅ Il tentativo di scrittura su una pipe piena è normalmente bloccante The write attempt on a full pipe is normally blocking
4. ✅ Una costante indica quanti byte possono essere scritti in una pipe in modo atomico A constant indicates how many bytes can be atomically written on a pipe
5. ⬜ L'operazione di scrittura su una pipe è sempre eseguita in modo atomico The write operation on a pipe is always performed atomically
6. ⬜ La lettura su una pipe in cui tutti gli estremi di scrittura sono stati chiusi provoca la generazione di un segnale di tipo SIGPIPE The read operation on a pipe, in which all writing extremes have been closed, causes the generation of a SIGPIPE signal

Exam 2021/09/07 - Ex 14 (1.5 points)

Italiano

Si supponga un processo esegua l'istruzione:

```
signal(SIGCHLD, SIG_IGN);
```

Si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose a process executes the following instruction:

```
signal(SIGCHLD, SIG_IGN);
```

Indicate which ones among 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. ✅ Il processo gestirà con la funzione di default SIG_IGN (definita tramite macro in signal.h) i segnali di tipo SIGCHLD. The process will handled signals of type SIGCHLD with the default function SIG_IGN (which is defined as a macro in signal.h).
2. ✅ Il processo non dovrà eseguire una wait o una waitpid. The process will not have to execute a wait or a waitpid.
3. ⬜ Il processo si comporterà in maniera standard alla ricezione di un SIGCHLD. The process will behave in a standard way when a SIGCHLD is received.
4. ✅ Il processo ignorerà segnali di tipo SIGCHLD. The process will ignore signals of type SIGCHLD.
5. ✅ Se il processo eseguirà una wait questa restituirà un codice di errore. If the process executes a wait, it will return an error code.
6. ⬜ Il processo figlio potrà diventare zombie. The child process can become zombie.

Exam 2020/09/14 - Ex 3 (2.0 points)

Italiano

Dati i segmenti di pseudo-codice inseriti al termine del testo in lingua inglese ed eseguiti in concorrenza dai processi P1 e P2 di PID pid_P1 e pid_P2, rispettivamente, indicare quali tra le seguenti affermazioni sono VERE.

English

Given the following segments of pseudo-code, which are executed concurrently by processes P1 and P2 with PID pid_P1 and pid_P2, respectively, indicate which of the following statements are TRUE.

```
P1
while (1) {
	...
	kill (pid_P2, SIGUSR1);
	pause ();
	A();
}

P2
while (1) {
	pause ();
	B();
	...
	kill (pid_P1, SIGUSR2);
}
```

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

1. ⬜ E' possibile l'esecuzione consecutiva della funzione A() per più di una volta (senza nessuna funzione B() nel mezzo) It is possible to execute function A() more than one time (without the execution of any B() function in between)
2. ⬜ La funzione A() può essere eseguita prima della funzione B() Function A() can be executed before function B()
3. ✅ Sono soggetti a deadlock They are subject to deadlocks
4. ✅ Sono soggetti a starvation They are subject to starvation
5. ⬜ La funzione B() è sicuramente eseguita almeno una volta Function B() is certainly executed at least one time
6. ✅ La funzione B() è sempre eseguita prima della funzione A() Function B() is always executed before function A()

Exam 2021/09/07 - Ex 9 (3.0 points)

Italiano
Si descriva che cosa è una pipe e come può essere utilizzata.
Perché è buona pratica chiudere i terminali inutilizzati di una pipe?
Si riporti il codice C che illustri come trasferire un insieme di stringhe di diversa dimensione, in modo che sia posssibile leggerle correttamente dall'altro lato della pipe.
English
Describe what is a pipe and how it can be used.
Why is it a good practice to close the unused terminals of a pipe?
Report a C code snippet to show how to transfer a set of strings of variable size, such that is is possible to correctly read them on the other side of the pipe.

Risposta: Answer:

Una pipe è un metodo di comunicazione tra processi. Può essere utilizzata per spedire dati o per la sincronizzazione tra processi. La pipe deve essere creata prima della fork per fare in modo che entrambi i processi, padre e figlio, possano accedere ai descrittori della pipe. Una volta creata, essa può essere utilizzato in modo simile ai file attraverso le system calls "read" e "write". La read è un operazione bloccante se la pipe è vuota. Se tutti i descrittori di file dei lettori sono chiusi ed è eseguita un'operazione di scrittura, un segnale di tipo SIGPIPE è spedito. Invece se un lettore cerca di leggere su una pipe in cui non esiste nessuno scrittore, è restituito il numero "0". Anche se le pipes sono un metodo di comunicazione di tipo half-duplex (cioè possono essere spedite informazioni in entrambe le direzioni, ma solo in una direzione per volta), è buona pratica chiudere uno dei due estremi (0 è tipicamente usato per la lettura, mentre 1 per la scrittura) al fine di evitare di scrivere in entrambi i lati della pipe allo stesso tempo. Per una comunicazione di tipo full-duplex dovrebbero essere utilizzate due pipe.
A pipe is an interprocess communication method. It can be used to send data and for synchronization between processes. The pipe must be created before the fork so that both parent and child can access the descriptors of the pipe. Once created, it can be used in a similar way to files with "read" and "write" system calls. Read is a blocking operation if the pipe is empty. If all the file descriptors of the readers are closed and a write operation is performed a SIGPIPE signal is sent, while if a reader tries to read from a pipe with no writers "0" is returned. Even if Pipes are a Half-duplex communication method (i.e., they could send information in each direction, but only one direction at the time) it is a good practice to close one of the two ends in its process (0 is commonly used for reading while 1 for writing) to avoid writing on both sides at the same time. For full-duplex communication, two pipes should be used.

```
#define N ...

int main () {
	int fd[2];
	char line[N];

	pipe (fd);

	if (fork()) {
		close (fd[0]);
		while (1) {
			scanf ("%s", line);
			write (fd[1], line, (strlen(line)+1)*sizeof(char));
		}
	} else {
		close (fd[1]);
		while (1) {
			read (fd[0], line, N*sizeof(char));
			printf ("%s\n", line);
		}
	}
	return (0);
```

```
}
```

Exam 2021/02/12 - Ex 8 (1.5 points)

Italiano

Si faccia riferimento all'utilizzo dei segnali in ambiente UNIX/Linux.
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Refer to the use of signals in a UNIX/Linux environment.
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. ⬜ 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
2. ✅ All'interno di un signal handler occorre usare solo funzioni rientranti Inside a signal handler, only reentrant functions should be used
3. ⬜ La ricezione di un segnale da parte di un processo provoca SEMPRE la terminazione del processo The reception of a signal by a process ALWAYS causes the termination of the process
4. ⬜ L'utilizzo della coppia di funzioni kill() e pause() o delle primitive semaforiche sem_post() e sem_wait() per la sincronizzazione dei processi sono EQUIVALENTI The use of the pair of functions kill() and pause() or of the semaphore primitives sem_post() and sem_wait() are EQUIVALENT from the point of view of process synchronization
5. ✅ La ricezione di alcuni tipi di segnali (ad esempio SIGKILL) non può essere ignorata The reception of some types of signal (for instance SIGKILL) cannot be ignored
6. ✅ L'esecuzione di un signal handler a seguito della ricezione di un segnale da parte di un processo può portare a race conditions The execution of a signal handler after the reception of a signal by a process can lead to race conditions

Exam 2021/06/18 - Ex 7 (1.5 points)

Italiano

Quali delle seguenti affermazioni sulle pipe sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Which of the following pipe 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. ✅ Le operazioni di scrittura su una pipe sono atomiche fino alla dimensione della costante PIPE_BUF. Write operations on a pipe are atomic up to the size of the constant PIPE_BUF.
2. ⬜ Un'operazione di lettura su una pipe è sempre bloccante. A read operation on a pipe is always blocking.
3. ⬜ Si può scrivere da ambo il lati della pipe contemporaneamente. It is possible to write on both sides of the pipe at the same time
4. ✅ Una pipe si riempie quando lo scrittore scrive troppo sulla pipe senza che il lettore legga nulla. A pipe gets filled up when the writer writes too much to the pipe without the reader reading any of it.
5. ✅ Un'operazione di scrittura è tipicamente bloccante su una pipe piena. A write operation is typically blocking on a full pipe.