**Esercitazione di laboratorio TPSIT #2**

**Cards**

classe 4 G – a.s. 2016/2017

**Carta vincente**

Scrivere un’applicazione in linguaggio C/Linux (cards.c) che consenta, ad un processo padre (croupier) di far competere due processi figlio (giocatori) nel gioco dell’estrazione della carta più alta.

Il croupier deve far estrarre una carta per volta a ciascun giocatore, rispettando sempre lo stesso ordine di chiamata (giocatore 1, giocatore 2 ecc..).

In modo più specifico, il croupier dovrà invitare un giocatore alla volta ad estrarre una carta; solo quando il giocatore l’avrà estratta, potrà invitare il secondo giocatore .. e così via.

I giocatori dovranno:

* estrarre (casualmente) una carta (per volta), da un ipotetico mazzo che ne contiene 14
* comunicare sempre al croupier la carta estratta

Il gioco dovrà ripetersi indefinitamente sino alla pressione della combinazione dei tasti “CTRL-C” allorchè il processo padre :

* invita i giocatori a lasciare il tavolo da gioco ( terminandoli )
* chiuderà il banco da gioco ( terminando a sua volta )

\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <signal.h>

// Firme dei metodi gestori

void GestoreSigInt(int ts);

void GestoreSigUsr1(int ts);

void GestoreSigUsr2(int ts);

// Gioco "Carta vincente"

int main(void)

{

int pid1, pid2,i ; // PID dei due giocatori

if (( pid1=fork() ) >0)

{

if (( pid2=fork() ) >0) // Croupier

{

signal(SIGINT, GestoreSigInt); // Registrazione gestore del croupier

printf("!Croupier !!!\n");

sleep(2); // Attesa per il setup dei giocatori

for (i=0; i<5; i++)

{

kill(pid1, SIGUSR1); // Chiedo al Giocatore 1 di giocare

pause();

kill(pid2, SIGUSR2); // Chiedo al Giocatore 2 di giocare

pause();

printf("\n");

}

printf("Croupier termina il gioco !!!\n");

printf("Invita i giocatori a lasciare il tavolo !!!\n");

kill ( pid1, SIGKILL ); // Terminazione dei giocatori

kill ( pid2, SIGKILL );

exit(0);

}

else

{ // Giocatore 2

signal(SIGUSR2, GestoreSigUsr2); // Registrazione gestore

for (;;)

{

srand(time(NULL));

pause();

printf(" Giocatore 2 --> Valore carta %d \n", 1+rand()%12);

sleep(1);

kill(getppid(), SIGINT); // Comunica al croupier di aver estratto la carta

}

}

}

else

{ // Giocatore 1

signal(SIGUSR1, GestoreSigUsr1); // Registrazione gestore

for (;;)

{

srand(time(NULL));

pause();

printf(" Giocatore 1 --> Valore carta %d \n", 1+rand()%12);

sleep(1);

kill(getppid(), SIGINT); // Comunica al croupier di aver estratto la carta

}

}

}

// Gestori dei segnali

void GestoreSigInt(int ts) { }

void GestoreSigUsr1(int ts) { }

void GestoreSigUsr2(int ts) { }