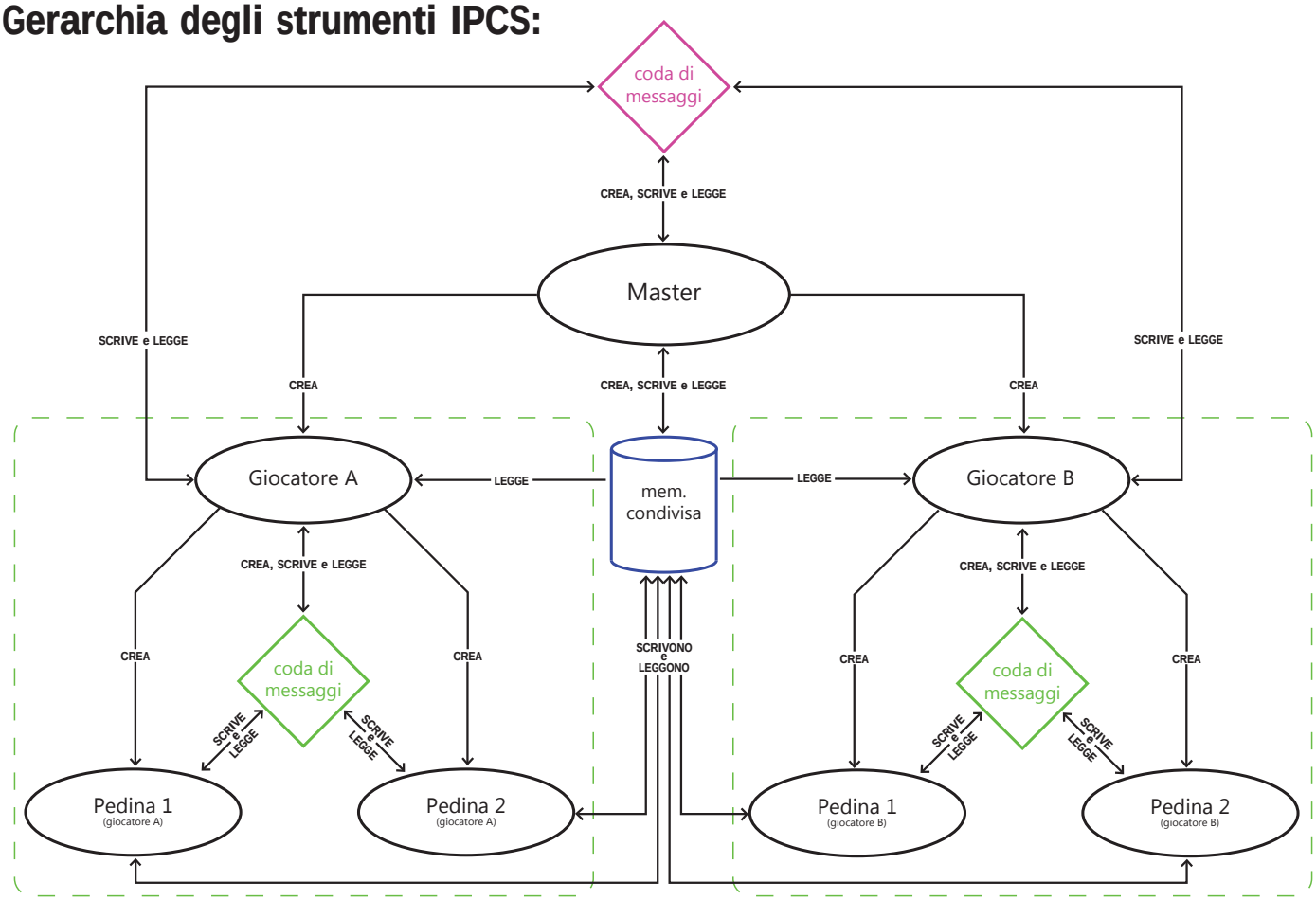
RELAZIONE PROGETTO DI SISTEMI OPERATIVI (2019-2020)

Samuel Girardello (n. matricola 886317) samuel.girardello@edu.unito.it
Marius Berinde (n. matricola 896991) marius.berinde@edu.unito.it

Gerarchia degli strumenti IPCS:



Strumenti di comunicazione:

- **Una memoria condivisa** per memorizzare la matrice che funge da scacchiera, sulla quale i processi eseguono operazioni di lettura e scrittura.
- **Una coda di messaggi** per far comunicare il processo master, i processi player e viceversa.
- **X code di messaggi** (con X uguale al numero di giocatori) per far comunicare i processi player con i processi pedina e viceversa.

Le code di messaggi vengono usate per trasferire informazioni di servizio tra i processi.

Sincronizzazione dei processi:

I processi vengono sincronizzati tramite delle code di semafori.

- **n.1 semaforo mutex** per salvaguardare l'accesso alla matrice.
- Il **processo master** ha **un semaforo dedicato alla sua sincronizzazione**. I processi player utilizzano questo semaforo per consentire a master la sua esecuzione.

- I **processi player** vengono gestiti dal **processo master** tramite una **coda di "n" semafori** dove "n" è il numero dei giocatori coinvolti nella partita. Master sincronizza i processi giocatore tramite questa coda di semafori.

- **Ogni processo player** ha **un semaforo** dedicato alla sua sincronizzazione. I processi pedina utilizzano questo semaforo per consentire a player la sua esecuzione.

- **I processi pedina** vengono gestiti dai rispettivi player tramite una **coda di "x" semafori** tanti quante sono le pedine per ogni giocatore. I giocatori sincronizzano i processi pedina tramite questa coda di semafori.

- Una **coda di "m" semafori,** tante quante sono le celle della scacchiera, è utilizzata per controllare se il percorso per raggiungere la destinazione è effettivamente percorribile.

Funzionamento dei semafori:

Vengono utilizzate 2 convenzioni diverse per la gestione dei semafori: **una è usata per la sincronizzazione dei processi e per mutex**, mentre l'altra **per la sincronizzazione delle pedine nella matrice.**

La convenzione utilizzata per la sincronizzazione dei processi è la seguente:
-Valore 0 indica che il processo ha il consenso a proseguire la sua esecuzione/accesso consentito
-Valore 1 indica che il processo sta in attesa dello "zero"/accesso non consentito
(Questa scelta è dovuta al fatto che usando una variabile di tipo struct sembuf e scegliendo di dare il valore 0 al campo sem_op permette di creare di creare una funzione simile alla wait vista a lezione).

Mentre per la sincronizzazione delle pedine è la seguente:
-Valore 0 indica che la cella della matrice è occupata da una pedina
-Valore 1 indica che la cella della matrice è libera, quindi non occupata da pedine

Tutti i semafori vengono inizializzati al valore 1. Ogni qualvolta si voglia dare il consenso ad un processo a eseguire la sua porzione di codice, il processo che sta eseguendo effettuerà una *semop* decrementando il valore di quel semaforo a 0. Il processo mandante attenderà quindi lo 0 per poter proseguire la sua esecuzione.

Osservazioni: Inizialmente, si erano usati i segnali (in specifico SIGCONT e SIGSTOP) unito alla waitpid(pid_processo, flag WSTOPPED) per sincronizzare i vari processi. A ricevimento venne però comunicato che non si poteva usare questo strumento per sincronizzare i processi, in quanto potrebbe interferire con determinate systemcall (come la nanosleep). Si è quindi dovuto riscrivere una parte di codice per poter implementare i semafori a sostituzione dei segnali.

Algoritmo di movimento

Una parte considerevole del lavoro è stata dedicata all'algoritmo di movimento. Le pedine si muovono sulla scacchiera solo con movimenti orizzontali o verticali (come la torre nel gioco degli scacchi).

I processi giocatore visitano la scacchiera calcolando la bandierina più vicina data dalla distanza tra la pedina a cui verrà data la destinazione e dalle bandierine presenti in scacchiera. Qualora la bandierina non si trovasse sulla stessa riga o sulla stessa colonna della pedina, viene calcolata una destinazione di "avvicinamento" calcolando prima se la strada tramite avvicinamento per riga è libera (ovvero non ci sono altre pedine sul percorso). Se l'avvicinamento non è possibile, si calcola se la strada tramite avvicinamento per colonna è libera. Qualora nessuna delle due strade sia percorribile, la pedina viene lasciata nella cella in cui si trova.

Regole da seguire per il corretto funzionamento del software

Durante lo sviluppo degli algoritmi sul posizionamento delle bandierine e sull'assegnamento del punteggio si è scelto di dare posizioni e punteggio in modo aleatorio. Si è però riscontrato che se il numero di bandiere è maggiore o uguale al 60% dello score totale allora non si riescono a creare degli score utili. Si è poi notato che **per avere una buona varietà nell'assegnamento dei punteggi bisogna che il numero totale delle bandiere debba essere inferiore al 50% dello score totale.**

Uso dell'utility make:

Per compilare il progetto digitare: **make compila**

Per il corretto funzionamento su sistemi OSX:

Commentare la struttura semun che si trova a capo del file "funzioni.h".