

# Sistemas de Computación

## 1er Semestre 2011

### Semáforos y Threads

Wenceslao Palma  
wenceslao.palma@ucv.cl

## Semáforos

Para trabajar con semáforos se debe seguir la siguiente secuencia: crear semáforo, operaciones sobre el semáforo (wait/signal) y destruir el semáforo. En forma más precisa, usando semáforos POSIX:

- Crear un semáforo. `int sem_init(sem_t *sem, int pshared, unsigned value);` donde `sem` es el semáforo, `pshared` es un argumento cuyo valor es 0 cuando el semáforo es compartido entre los threads de un proceso y `value` es el valor inicial del semáforo.
- Operación wait. `int sem_wait(sem_t *sem);`
- Operación signal. `int sem_post(sem_t *sem);`
- Destruir un semáforo. `int sem_destroy(sem_t *sem);`

```
#include <semaphore.h>

sem_t semaforo;
main(){
    .....
    sem_init(&semaforo,0,2);
    sem_wait(&semaforo);
    printf("estoy en la sección crítica \n");
    sem_post(&semaforo);
    .....
}
```

## Threads

Para crear un thread se utiliza la función

`int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine)(void*), void *arg);` donde `thread` identifica al thread en creación, `attr` son los atributos del thread, `start_routine` es la función que ejecutará el thread y `arg` son los argumentos de `start_routine`. Importante: todos los threads comparten variables gloables.

```
#include <pthread.h>

void *print(void *threadID)
{
    long tid;
    tid = (long)threadID;
    printf("thread [##%ld]: hola!\n", tid);
    pthread_exit(NULL);
}

void main ()
{
    pthread_t thread1,thread2;
    int id1=1,id2=2;

    printf("creando thread 1\n");
    pthread_create(&thread1, NULL, print, (void *)id1);

    printf("creando thread 2\n");
    pthread_create(&thread2, NULL, print, (void *)id2);

    pthread_exit(NULL);
}
```