Process Management Example

ExecHandler (char* executable)
Process* p = new Process();
Process* parent = CurrentProcess();
create and initialize VAS for p;
p->Birth(parent);
start thread in child context;
allocate and return SpaceID;

JoinHandler(int spaceid)
Process* child = get from spaceID;
int status = child->Join();
return(status);

ExitHandler(int status)
Process* p = CurrentProcess();
tear down p’s VAS, release memory;
p->Death(status);
delete p;
destroy this thread;

1. Parent waits in Join for child to exit (by calling Death).
2. Exited child waits in Death for parent to join or exit.
3. Exited parent waits in Death for children to exit.

Needed:
• 3 wait
• 3 signal/broadcast each in (Join, Death, Death)
Process Birth Example
(Midterm Problem #4)

Mutex* pMx;
Condition* pCv;

void
Process::Birth(Process *parent) {
  pMx->Acquire();
  this->parent = parent;
  parent->AddChild(this);
  status = 0;
  exited = 0;
  joinedOn = 0;
  pMx->Release();
}

*pMx and pCv are global
(coULD USE PRIVATE CVS, BUT IT’S TRICKY)
Process Join Example
(Midterm Problem #4)

```c
int Process::Join() {
    int status;

    pMx->Acquire();
    while (!exited)
        pCv->Wait(); /* wait for child to exit */
    status = this->status;
    joinedOn = 1;
    pCv->Broadcast(); /* tell child that parent is done */
    pMx->Release();
    return(status);
}
```

1. Parent waits in Join for child to call Death.
2. Child waits in Death for parent to complete Join.

Common errors:
(1) What if I don’t kick the condition variable? Child may block in Death() waiting for parent to Join.
(2) What if I set joinedOn too soon, before waiting? Child may return from Death thinking the join is complete.
(3) What if release the lock without saving status in a local variable? Child may return from Death and delete its process object as soon as I tell it the Join is complete.
Process Death Example

```c
void Process::Death(int status) {
    pMx->Acquire();

    exited = 1;           /* process has exited */
    this->status = status;
    pCv->Broadcast(); /* wake parent from join */

    while (!joinedOn && !parent->exited)
        pCv->Wait();
    parent->RemoveChild(this); /* parent may delete now */
    pCv->Broadcast();

    while (!children->Empty())
        pCv->Wait();

    pMx->Release();
}
```

1. Parent waits in `Join` for child to call `Death`.
2. Child waits in `Death` for parent to call `Death`.
3. Parent waits in `Death` for children to call `Death`.

Common errors:

(1) What if I wait for children before telling parent I exited? Parent must wait for all descendants to exit before it can reap child status and continue from `Join`.

(2) What if I RemoveChild too early, before verifying that parent has joined or exited? Parent may return from `Death` and delete itself, before I have finished looking at its exited flag.

(3) If I wait for parent to exit or complete join before telling it I exited, then I may deadlock if the parent is waiting for me in `Join`.