Sistemas Operativos

Implementación de mensajes en nThread

Diego Madariaga

1.
Mensajes

Mensajes

- `int nSend(nThread th, void *msg);`

  Envía un mensaje msg al thread th. Suspende la ejecución hasta recibir una respuesta de parte de th. Retorna el valor recibido

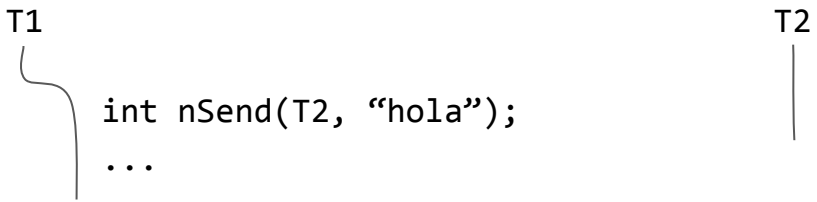
- `void *nReceive(nThread *pth, int timeout_ms);`

  Suspende la ejecución hasta recibir un mensaje. Escribe el descriptor del thread que envió el mensaje en *pth. Retorna el mensaje recibido. Si timeout es mayor a 0, retornará de todas formas luego de esa cantidad de tiempo.

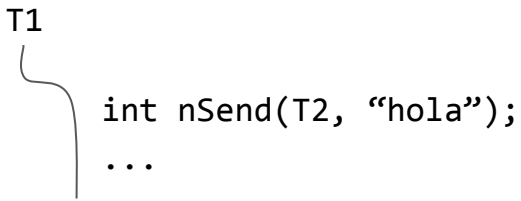
- `void nReply(nThread th, int rc);`

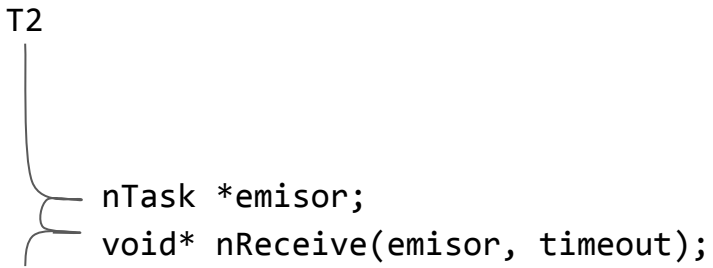
  Responde a un mensaje recibido de parte de th con el código de retorno rc. No suspende la ejecución.

Mensajes

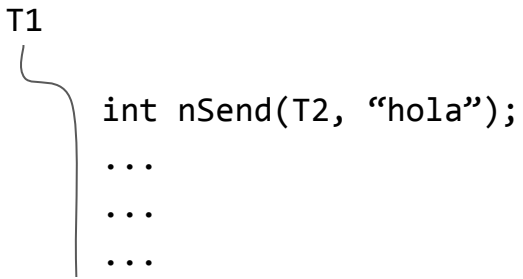


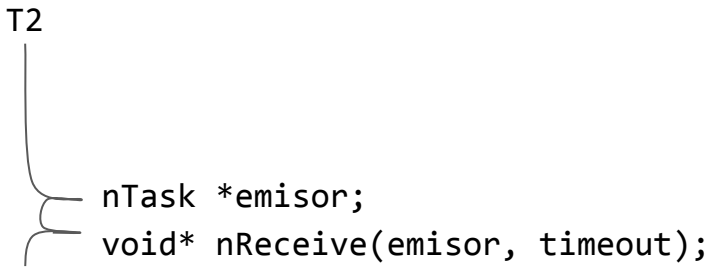
Mensajes


T1
int nSend(T2, “hola”);
...

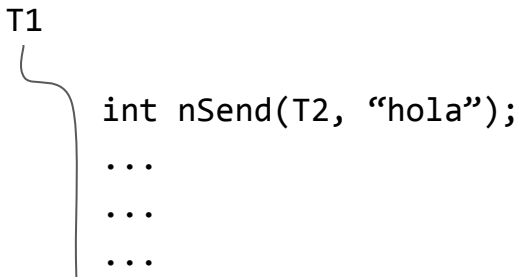

T2
nTask *emisor;
void* nReceive(emisor, timeout);

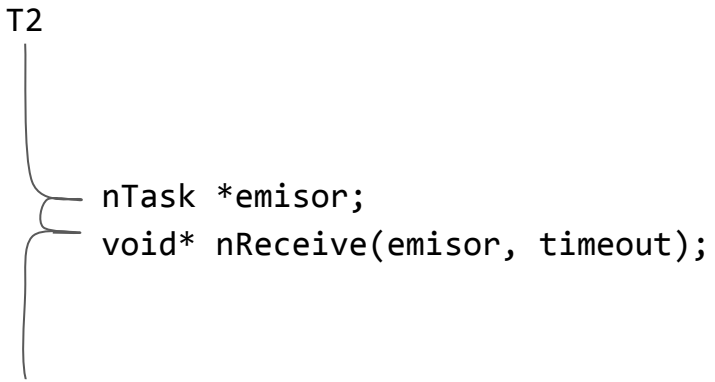
Mensajes


T1
int nSend(T2, “hola”);
...
...
...

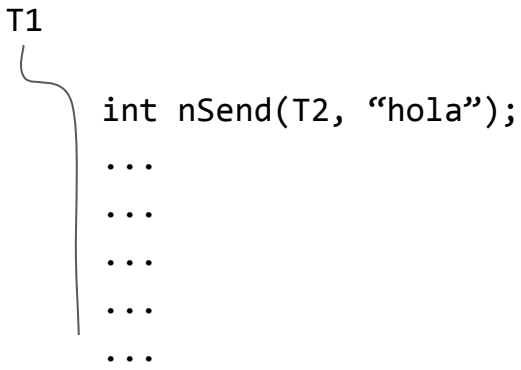

T2
nTask *emisor;
void* nReceive(emisor, timeout);

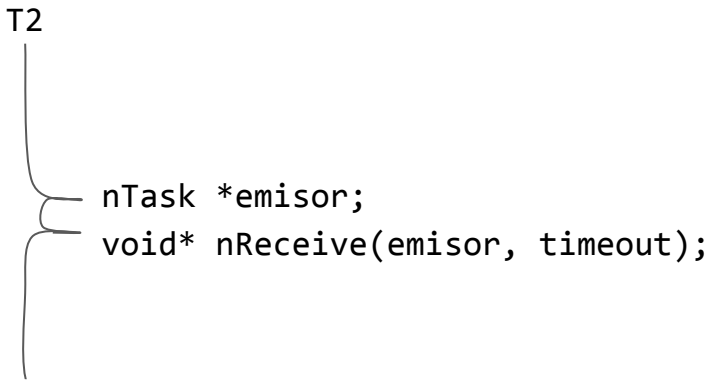
Mensajes


T1
int nSend(T2, “hola”);
...
...
...

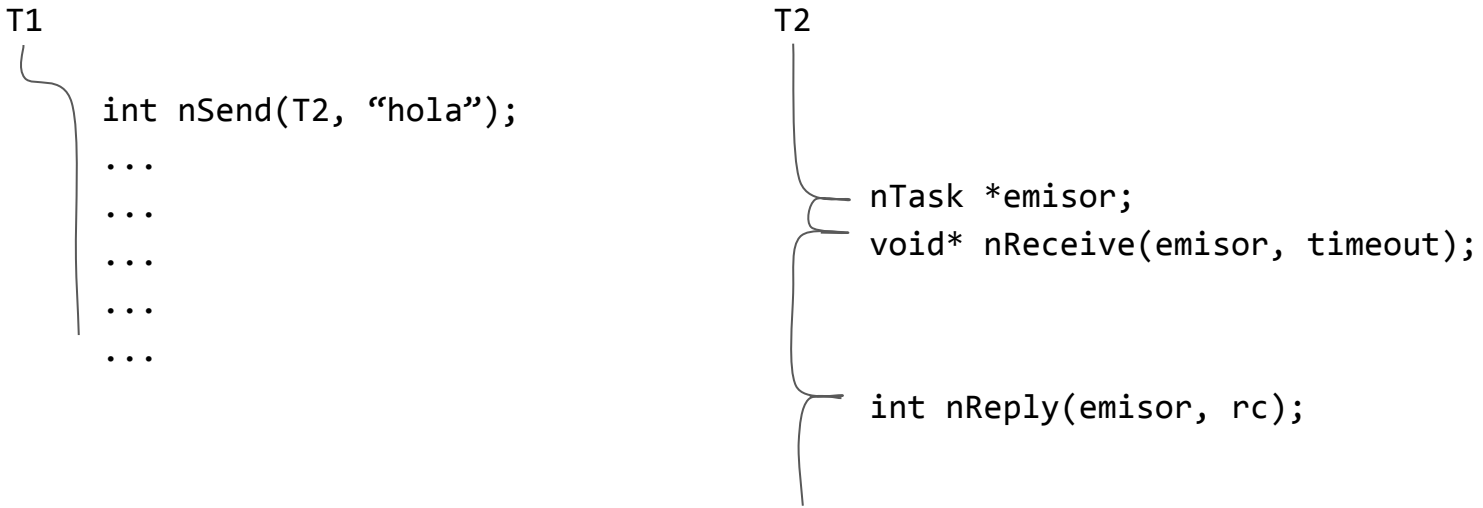

T2
nTask *emisor;
void* nReceive(emisor, timeout);

Mensajes

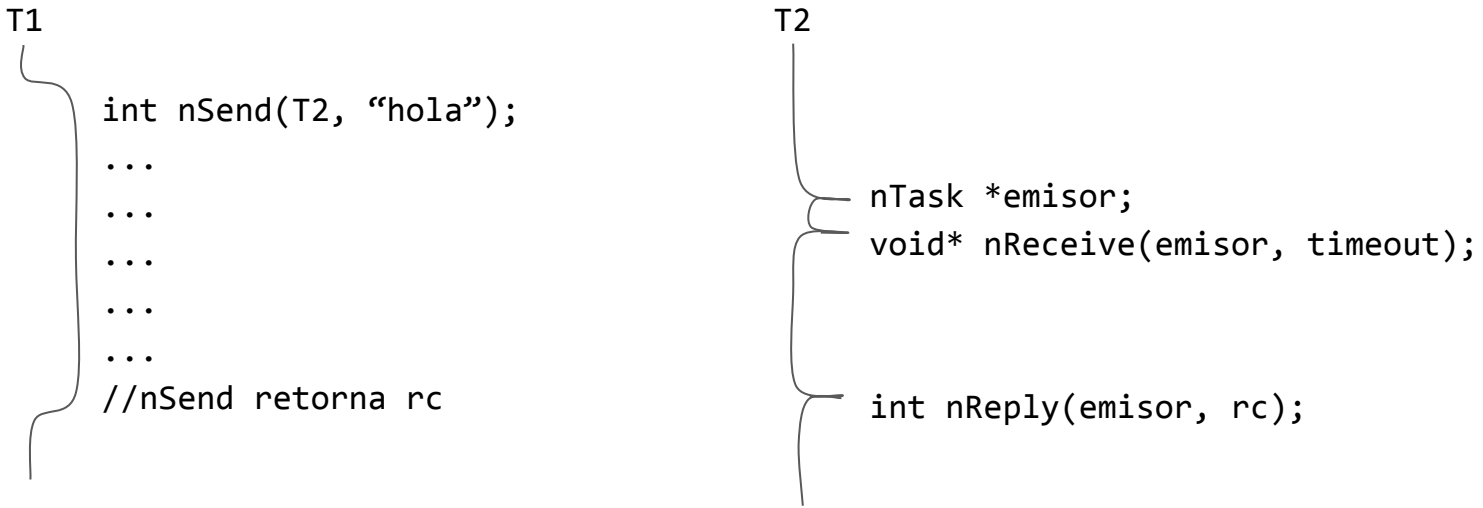

T1
int nSend(T2, “hola”);
...
...
...
...
...


T2
nTask *emisor;
void* nReceive(emisor, timeout);

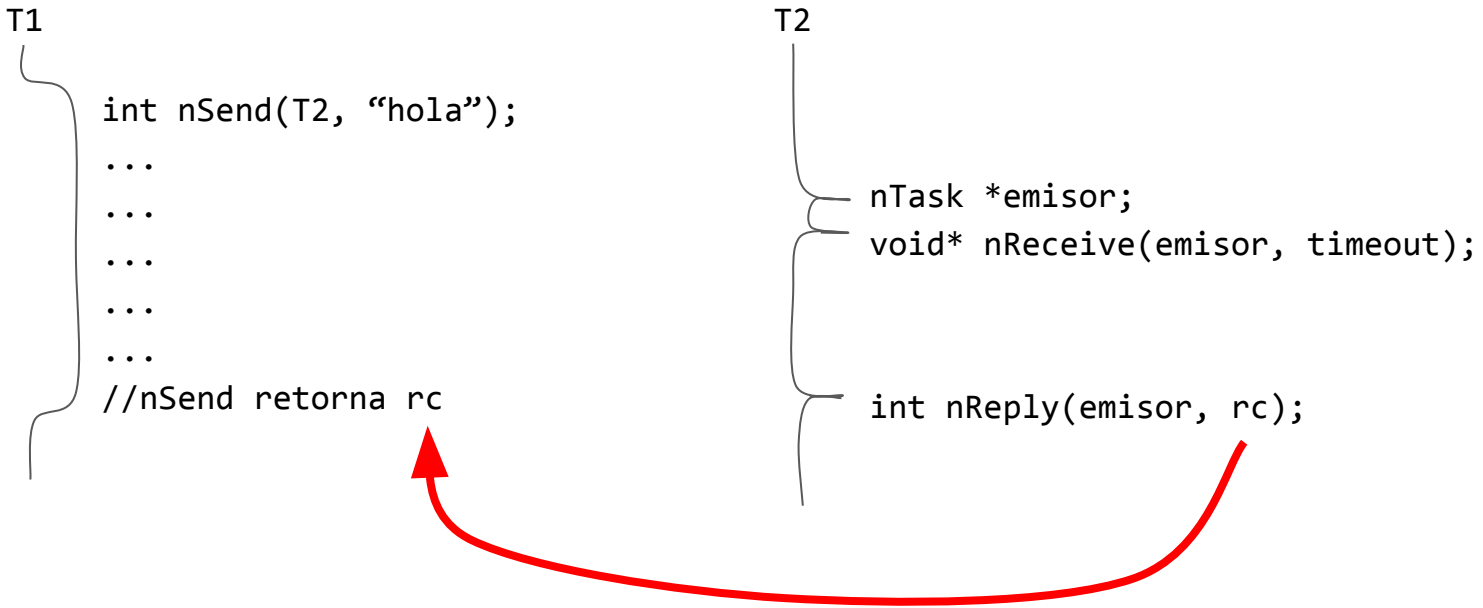
Mensajes


T1
int nSend(T2, “hola”);
...
...
...
...
...
T2
nTask *emisor;
void* nReceive(emisor, timeout);
int nReply(emisor, rc);

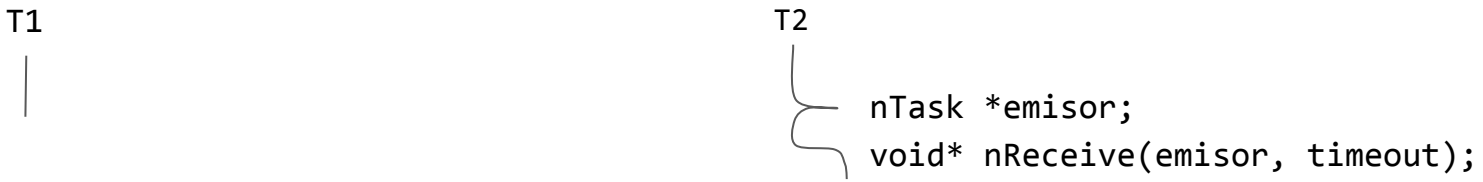
Mensajes


T1
int nSend(T2, “hola”);
...
...
...
...
...
//nSend retorna rc
T2
nTask *emisor;
void* nReceive(emisor, timeout);
int nReply(emisor, rc);

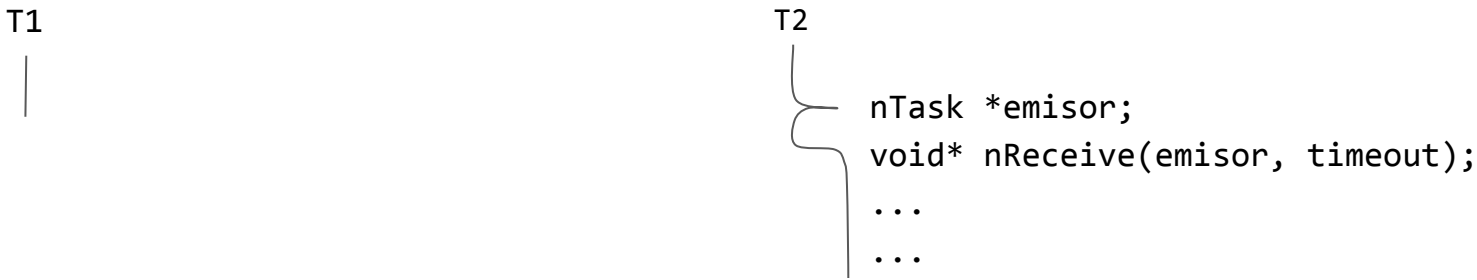
Mensajes


T1
int nSend(T2, “hola”);
...
...
...
...
...
//nSend retorna rc
T2
nTask *emisor;
void* nReceive(emisor, timeout);
int nReply(emisor, rc);

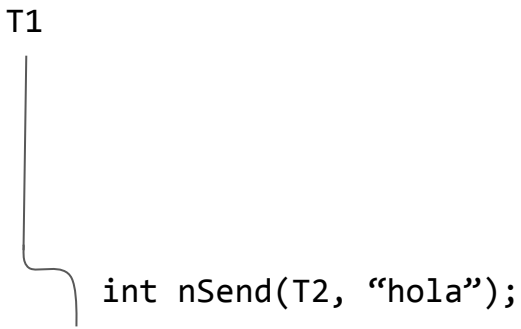
Mensajes

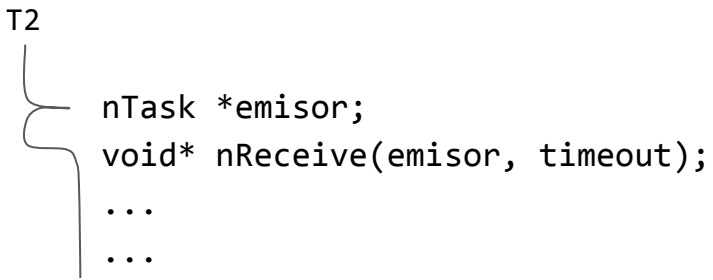


Mensajes

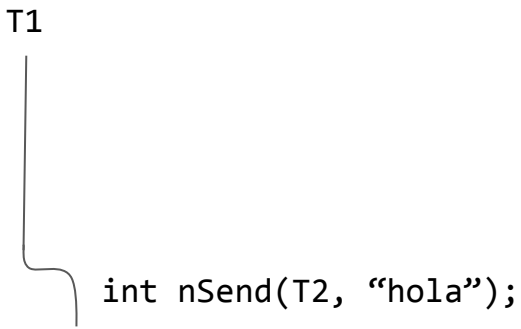


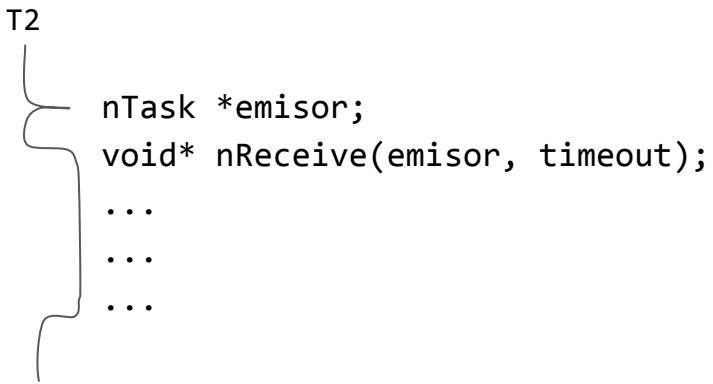
Mensajes


T1
int nSend(T2, “hola”);

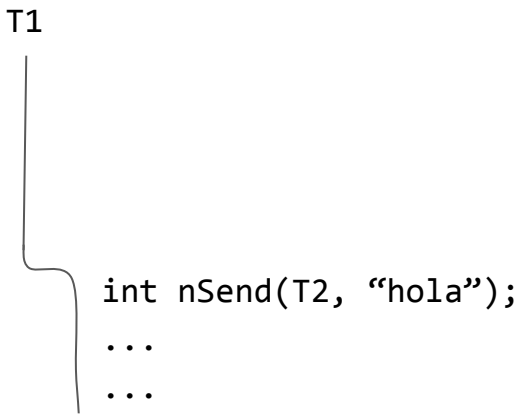

T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...

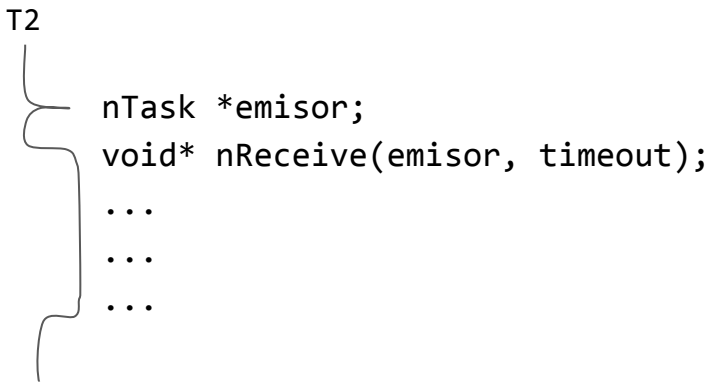
Mensajes


T1
int nSend(T2, “hola”);

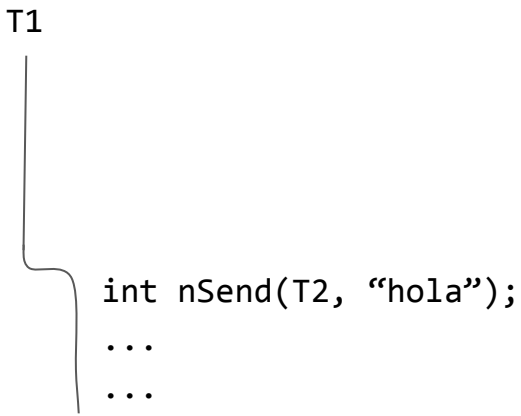

T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...
...

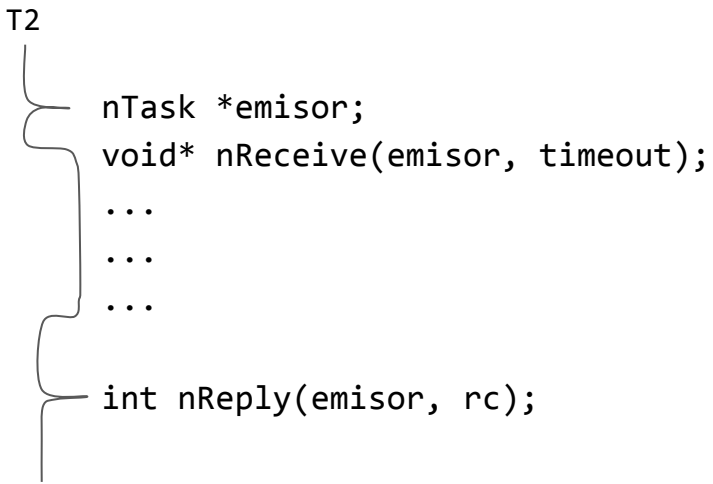
Mensajes


T1
int nSend(T2, “hola”);
...
...

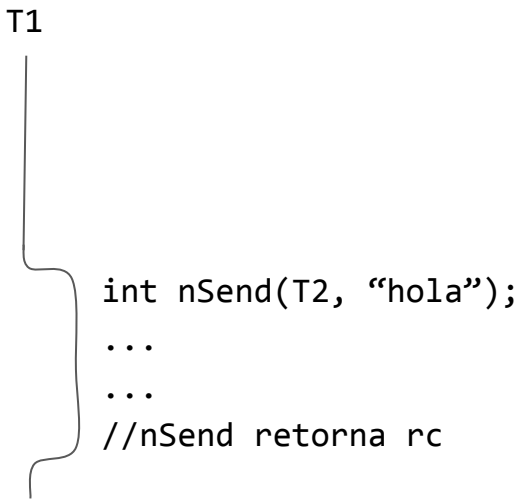

T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...
...

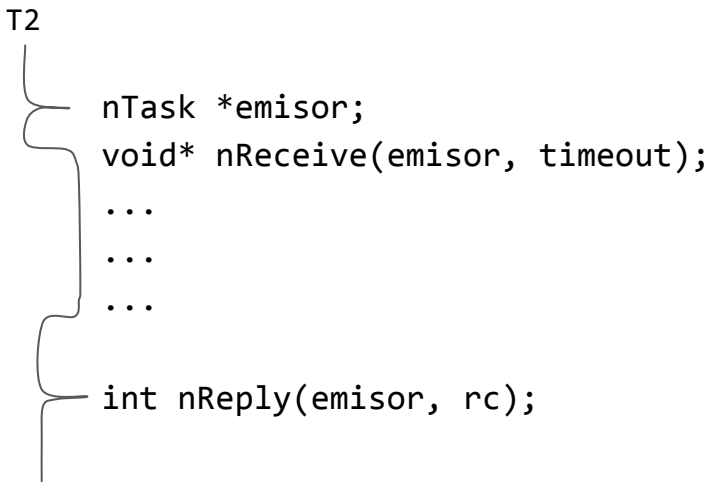
Mensajes


T1
int nSend(T2, “hola”);
...
...

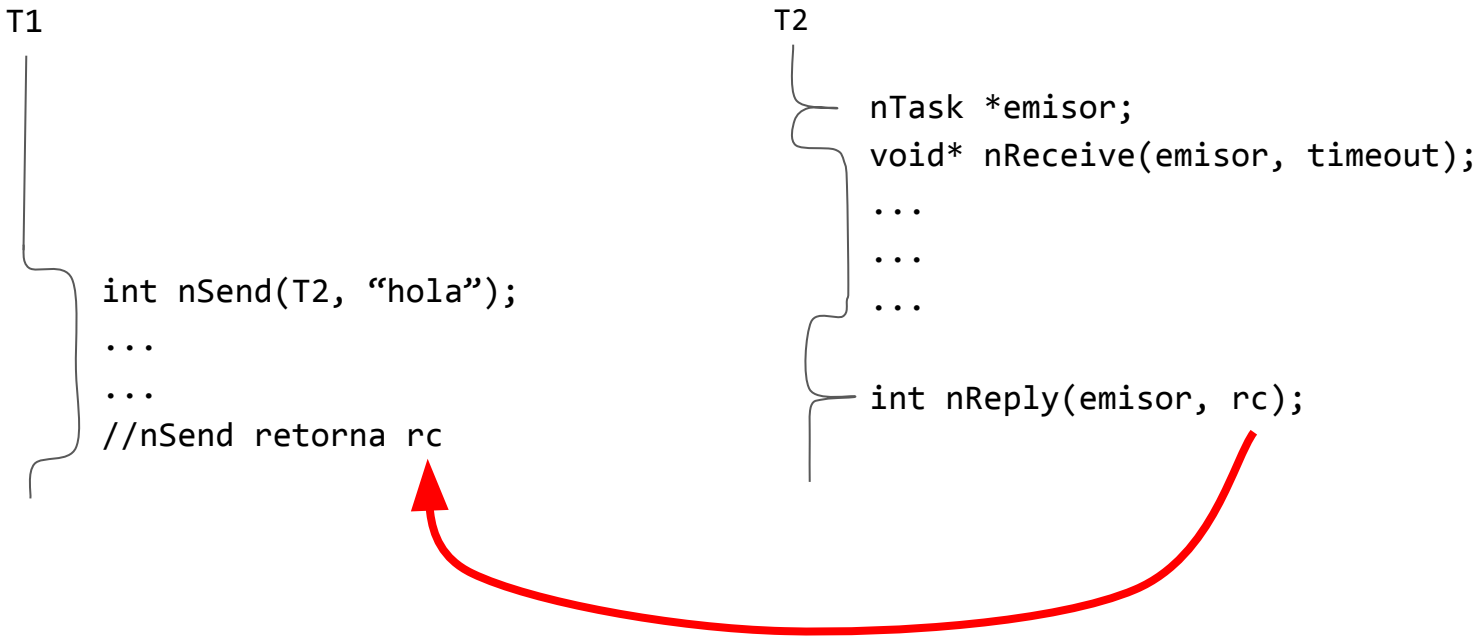

T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...
...
int nReply(emisor, rc);

Mensajes

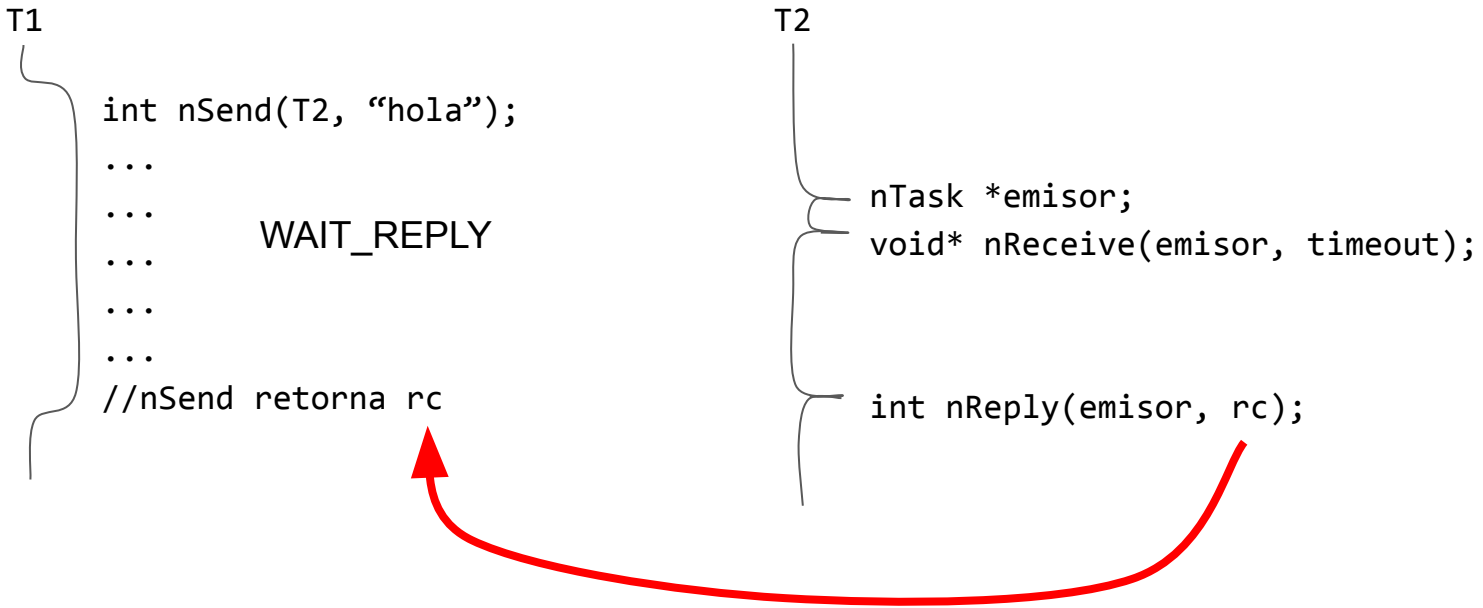

T1
int nSend(T2, “hola”);
...
...
//nSend retorna rc


T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...
...
int nReply(emisor, rc);

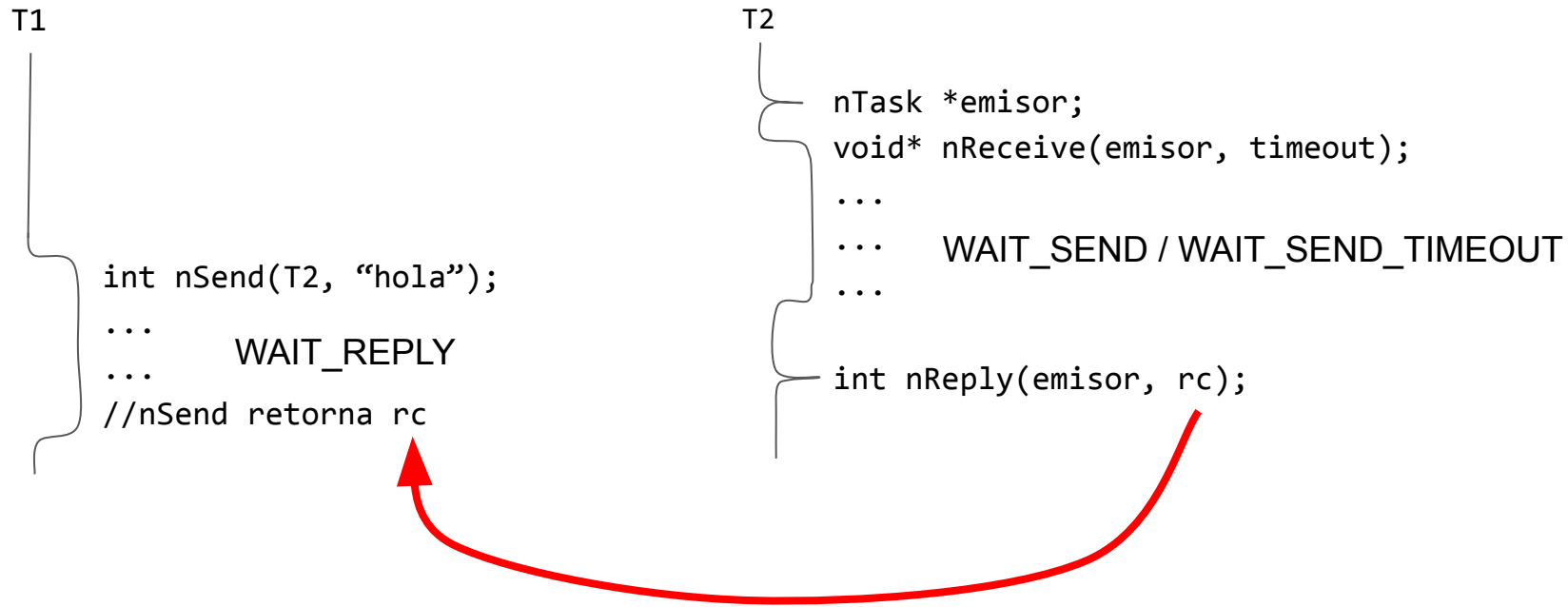
Mensajes


T1
int nSend(T2, “hola”);
...
...
//nSend retorna rc
T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...
...
int nReply(emisor, rc);

Mensajes


T1
int nSend(T2, “hola”);
...
...
...
...
...
//nSend retorna rc
WAIT_REPLY
T2
nTask *emisor;
void* nReceive(emisor, timeout);
int nReply(emisor, rc);

Mensajes


T1
int nSend(T2, “hola”);
...
...
//nSend retorna rc
WAIT_REPLY
T2
nTask *emisor;
void* nReceive(emisor, timeout);
...
...
...
WAIT_SEND / WAIT_SEND_TIMEOUT
int nReply(emisor, rc);

2.
Ejemplo de uso: Impresora Compartida

3.
Implementación en nThreads

Sistemas Operativos

Implementación de mensajes en nThread

Diego Madariaga