IPC with PHP : API, Approaches and Applications

Andrey Hristov
100 DAYS – Software Projects

LinuxTag 2004
June 23rd, 2004

About me

- BSc in Computer Engineering
- Student at University of Applied Sciences, Stuttgart, Germany
- Programming for the Web since year 2000
- Working on the PHP core since year 2002
- Author of pecl/stats
- Working for 100 Days GmbH, Germany

Survey

How many of you know about IPC in means of shared memory, semaphores, message queues?

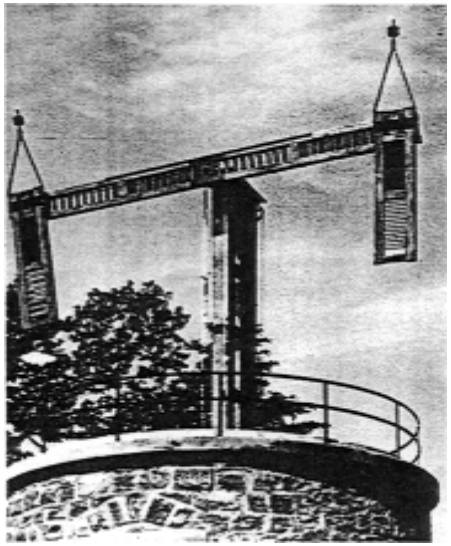
How many of you have worked with these in PHP?

Agenda

- Semaphores
- Shared memory
- Message queues
- PHP API
- pecl/memcache
- Procedural examples of IPC primitives
- OO approach to IPC
- Use cases

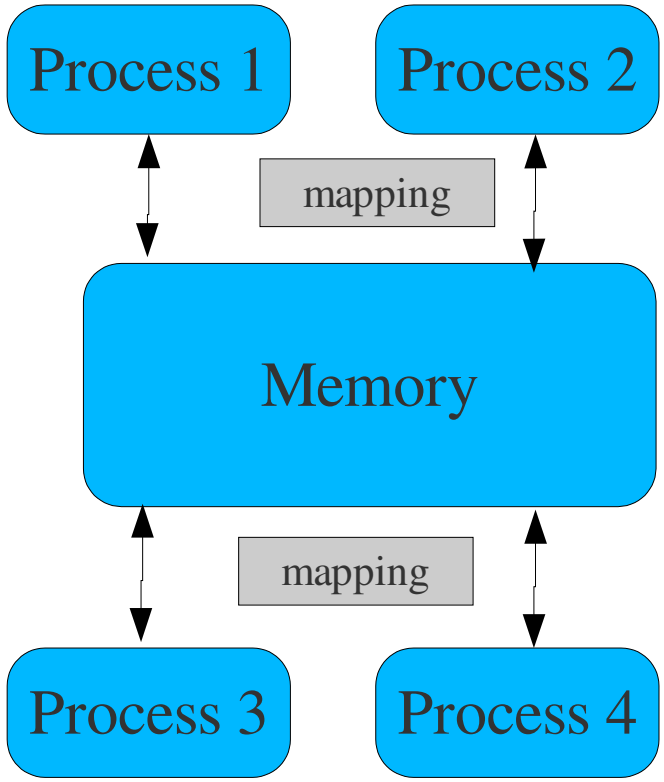
Semaphores

- Invented by Dijkstra . P(s) and V(s) operations
- Used for synchronization
- Binary semaphores
- Non-binary semaphores



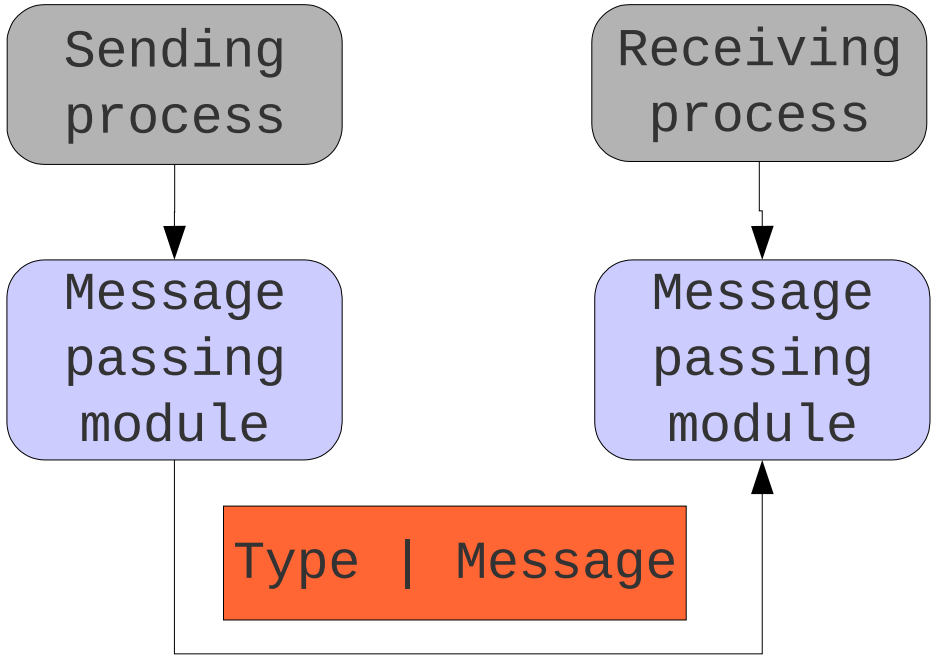
Shared memory

- Memory segments available for more than one process.
- The processes map their virtual space to the shared memory segment.
- In System V implementations the access is permision based.
- Synchronization is needed in cases of mixed reads/writes. Binary semaphore is usually used then.



Message queues

- Simple way of data exchange: send messages.
- IPC messaging allows sending and receiving in arbitrary order – on the contrary of reading from a stream. One can fetch element from the queue even if it's not the first one.
- Blocking and non-blocking receival.



PHP API for System V IPC (0)

- ext/sysvshm --enable-sysvshm (>= 3.0.6)
- ext/sysvsem --enable-sysvsem (>= 3.0.6)
- ext/sysvmsg --enable-sysvmsg (>= 4.3.0)

PHP API for System V IPC (1)

- shm_attach()
- shm_detach()
- shm_get_var()
- shm_put_var()
- shm_remove_var()
- shm_remove()
- sem_acquire()
- sem_get()
- sem_release()
- sem_remove()
- msg_get_queue()
- msg_receive()
- msg_remove_queue()
- msg_send()
- msg_set_queue()
- msg_stat_queue()

ext/shmop

- Written by Ilia Alshanetsky & Slava Poliakov
- Works on Win32 and *nix
- Better interoperability with the non-PHP world.
- Since PHP 4.0.3. The current naming is since 4.0.4 .
- shmop_open()
- shmop_close()
- shmop_delete()
- shmop_read()
- shmop_write()
- shmop_size()

pecl/memcache

- “memcached is a high-performance, distributed memory object caching system, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load”.
- memcached runs on every server in the web server farm. It is CPU lightweight and memory “hungry”. Good symbiosis with httpd.
- Works on *nix and Win32 (PHP4 and PHP5).
- Both procedural and OO API.
- Easy to install : “pear install memcache”

Working with semaphores (procedural way) (0)

```
<?php
$key = ftok(__FILE__, 'R'); var_dump($key);
$sem_id = sem_get($key, 1, 0666);
if ($argc == 2 && $argv[1] == "clear") {
        echo "Clearing semaphore : $key\n";
        sem_remove($sem_id); exit;
}
$res = sem_acquire($sem_id);
var_dump(strftime('TIME :  %H:%M:%S', (int)microtime(1)));
sleep(5);
$res = sem_release($sem_id);
var_dump(strftime('TIME :  %H:%M:%S', (int)microtime(1)));
?>
```

```
andrey@poohie:~/test/php> php sem_create.php
int(1376196715)
string(16) "TIME :  16:12:05"
string(16) "TIME :  16:12:10"
```

```
andrey@poohie:~/test/php> php sem_create.php
int(1376196715)
string(16) "TIME :  16:12:10"
string(16) "TIME :  16:12:15"
```

Working with shared memory (procedural way) (0)

```
<?php
$key = ftok(__FILE__, 'R'); var_dump($key);
$shm_id = shm_attach($key, 1024, 0666);
if ($argc == 2 && $argv[1] == "clear") {
        echo "Clearing shared memory : $key\n";
        shm_detach($shm_id); exit;
}
$var = strftime('TIME :  %H:%M:%S', (int)microtime(1));
echo "Putting into SHM : ";
var_dump($var);
$res = shm_put_var($shm_id, 1/*var_key*/, $var);
var_dump($res);
$var2 = shm_get_var($shm_id, 1/*var_key*/);
echo "Read from SHM : ";
var_dump($var2);
shm_remove_var($shm_id, 1);
?>
```

```
andrey@poohie:~/test/php> php shm_create.php
int(1376196715)
Putting into SHM : string(16) "TIME :  17:17:54"
bool(true)
Read from SHM : string(16) "TIME :  17:17:54"
```

Working with shared memory (procedural way) (1)

```
<?php
$key = ftok('/home/andrey/test/phpconf/shm_create2.php', 'R');
$shm_id = shm_attach($key, 1024, 0666);
$var = strftime('TIME :  %H:%M:%S', (int)microtime(1));
echo "Putting into SHM : ";
var_dump($var);
$res = shm_put_var($shm_id, 1/*var_key*/, $var);
var_dump($res);
sleep(10);
$var2 = shm_get_var($shm_id, 1/*var_key*/);
echo "Read from SHM : ";
var_dump($var2);
?>
```

```
andrey@poohie:~/test/php> php shm_create2.php
Putting into SHM : string(16) "TIME :  17:54:47"
bool(true)
Read from SHM : string(16) "TIME :  17:54:51"
```

```
andrey@poohie:~/test/php> php shm_create2.php
Putting into SHM : string(16) "TIME :  17:54:51"
bool(true)
Read from SHM : string(16) "TIME :  17:54:51"
```

Working with shared memory (procedural way) (2)

```
<?php
$key = ftok(__FILE__, 'R');
$shm_id = shm_attach($key, 1024, 0666);
$sem_id = sem_get($key, 1, 0666);
if ($argc == 2 && $argv[1] == "clear") {
  echo "Clearing shared memory / semaphore : $key\n";
  shm_detach($shm_id);sem_remove($sem_id);exit;
}
$var = strftime('TIME :  %H:%M:%S', (int)microtime(1));
echo "Putting into SHM : ";var_dump($var);
sem_acquire($sem_id);
printf("Acquired access @%s:\n", strftime('TIME :  %H:%M:%S', (int)microtime(1)););
$res = shm_put_var($shm_id, 1/*var_key*/, $var);
sleep(10); // sleep 10 seconds
$var2 = shm_get_var($shm_id, 1/*var_key*/);
sem_release($sem_id);
echo "Read from SHM : ";var_dump($var2);
?>
```

```
andrey@poohie:~/test/php> php shm_create3.php
Putting into SHM : string(16) "TIME :  18:03:23"
Acquired access @TIME :  18:03:23:
Read from SHM : string(16) "TIME :  18:03:23"
```

```
andrey@poohie:~/test/php> php shm_create3.php
Putting into SHM : string(16) "TIME :  18:03:25"
Acquired access @TIME :  18:03:33:
Read from SHM : string(16) "TIME :  18:03:25"
```

Working with a message queue

Server :

```
<?php
$key = ftok("/home/andrey/test/phpconf/mqueue.php", 'R');
$queue = msg_get_queue($key, 0666);
$message = NULL;
$err = NULL;
msg_receive($queue, 1/*desired*/, $real_type, 16384, $message, 1/*ser*/,0,$err)
echo "Received : ";var_dump($message);
echo "Error code : ";var_dump($err);
?>
```

Client :

```
<?php
$key = ftok("/home/andrey/test/phpconf/mqueue.php", 'R');
$queue = msg_get_queue($key, 0666);
$message = "Hello World!";
$err = 0;
echo "Sending :";var_dump($message);
msg_send($queue, 1/*msg_type*/, $message, 1/*ser*/, TRUE,$err);
var_dump($err);
?>
```

```
andrey@poohie:~/test/php> php mqueue_client.php
Sending :string(12) "Hello World!"
int(0)
```

```
andrey@poohie:~/test/php> php mqueue.php
Received : string(12) "Hello World!"
Error code : int(0)
```

ext/shmop example (0)

```
<?php
// Create 100 byte shared memory block
$shm_id = shmop_open(ftok(__FILE__,'R'), "c"/*mode*/, 0644/*rights*/, 100/*size*/);
if (!$shm_id) {
     echo "Couldn't create shared memory segment\n";exit;
}
// Get shared memory block's size
$shm_size = shmop_size($shm_id);
echo "SHM Block Size: " . $shm_size . " has been created.\n";
// Lets write a test string into shared memory
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if ($shm_bytes_written != strlen("my shared memory block")) {
     echo "Couldn't write the entire length of data\n";exit;
}
// Now lets read the string back
$my_string = shmop_read($shm_id, 0/*start*/, $shm_size/*count*/);
if (!$my_string) { echo "Couldn't read from shared memory block\n"; exit; }
echo "The data inside shared memory was: " . $my_string . "\n";
//Now lets delete the block and close the shared memory segment
if (!shmop_delete($shm_id)) {
     echo "Couldn't mark shared memory block for deletion.";exit;
}
shmop_close($shm_id);
?>
```

```
andrey@poohie:~/test/php> php shmop.php
SHM Block Size: 100 has been created.
The data inside shared memory was: my shared memory block
```

ext/shmop example (1) : semaphore emulation

```
<?php
$key = ftok(__FILE__, "K");
error_reporting(0);
do {
     $sem_id = shmop_open($key, "n", 0644, 10);
     usleep(100);
} while ($sem_id === false);
error_reporting(E_ALL);
echo "Entered critical section at :".strftime("%H:%M:%S\n", time());
sleep(5);
shmop_delete($sem_id);
echo "Exited critical section at  :".strftime("%H:%M:%S\n", time());
?>
```

```
andrey@poohie:~/test/phpconf> php shmop2.php
Entered critical section at :16:50:41
Exited critical section at  :16:50:46
```

```
andrey@poohie:~/test/phpconf> php shmop2.php
Entered critical section at :16:50:46
Exited critical section at  :16:50:51
```

pecl/memcache example

```
<?php
$mc = memcache_connect('localhost', 11211);
if ($mc) {
     $mc->set("num_key", 123);
     $mc->set("str_key", "String to store in memcached");
     $mc->set("arr_key", array('assoc'=>123, 345, 567));

     $object = new stdClass;
     $object->attribute = 'test';
     $mc->set("obj_key", $object);

     var_dump( $mc->get('str_key'),
               $mc->get('num_key'),
               $mc->get('obj_key'),
               $mc->get('arr_key')
          );
} else {
     echo "Connection to memcached failed";
}
?>
```

```
andrey@poohie:~> php example.php
string(28) "String to store in memcached"
string(3) "123"
object(stdClass)#3 (1) {
  ["attribute"]=>
  string(4) "test"
}
array(3) {
  ["assoc"]=>
  int(123)
  [0]=>
  int(345)
  [1]=>
  int(567)
}
```

Using semaphores/shared memory in a OO way

- Procedural way is makes the code larger and more error prone.
- With OO semaphore, one has only the P and V operations visible to the programmer :
  - *acquire()*
  - *release()*.
- With OO shared memory, one uses only
  - *getVar()*
  - *putVar()*
- The parameters are passed during instantiation.

OO semaphores : example

```
<?php
 require 'shm.php';
 $sem = new Shm_Semaphore("test", 2); // semaphore's name is "test"
                   // max_acquire is 2. All processes can use
                   // the semaphore
 $sem->acquire();
 printf("Entered section at : %s\n", strftime("%D %T", time()));
 sleep(5);
 printf("Exited  section at : %s\n", strftime("%D %T", time()));
 $sem->release();
?>
```

```
andrey@poohie:~/test/phpconf> php sem1.php
Entered section at : 04/25/04 17:00:32
Exited  section at : 04/25/04 17:00:37
```

```
andrey@poohie:~/test/phpconf> php sem1.php
Entered section at : 04/25/04 17:00:33
Exited  section at : 04/25/04 17:00:38
```

OO shared memory : example

```
<?php
 require 'shm.php';
 //allocating 1024 bytes for the variable
 $shm_var = new Shm_Var("acm3", 1024, "666");
 if (!($str = $shm_var->getVar())) {
   $str = strftime("First script started at : %H:%M:%S\n", time());
   $shm_var->putVar($str);
 }
 echo "From memory : ".$str;
 echo "Current time: ".strftime("%H:%M:%S\n", time());
?>
```

```
andrey@poohie:~/test/phpconf> php shm1.php
From memory : First script started at : 17:23:58
Current time: 17:23:58
```

```
andrey@poohie:~/test/phpconf> php shm1.php
From memory : First script started at : 17:23:58
Current time: 17:24:01
```

OO protected shared memory : the class

```
<?php
class Shm_Protected_Var {
  var $_debug       = false;
  var $_sem         = NULL;
  var $_shm_var     = NULL;
  var $_cached_val  = NULL;
  var $_in_section  = false;

  function Shm_Protected_Var($name, $size) {
    $sname = substr($name,0,4);
    $this->_sem=&new Shm_Semaphore($sname,1);
    $this->_shm_var=&new Shm_Var($sname,$size);
  }

  function startSection() {
    if ($this->_in_section === true) {
      printf("Already in critical section\n");
      return false;
    }
    $this->_sem->acquire();
    $this->_in_section = true;

    return true;
  }
  function endSection() {
    if ($this->_in_section === false) {
      printf("Not in critical section\n");
      return false;
    }
    $this->_in_section = FALSE;
    $this->_sem->release();

    return true;
  }
  function setVal($val) {
    if ($this->_in_section === false) {
      printf("Not in critical section\n");
      return false;
    }
    $this->_cached_val = $val;
    $this->_shm_var->putVar($this->_cached_val);
    return true;
  }
  function getVal() {
    if ($this->_in_section === false) {
      printf("Not in critical section\n");
      return false;
    }
    return $this->_shm_var->getVar();
  }
} ?>
```

OO protected shared memory: example

```
<?php
  require 'shm.php';

  $guarded_var = new Shm_Protected_Var("acm4", 1024);//allocating 1024 bytes
  $guarded_var->startSection();
  $s = sprintf("putVal() at : %s\n",microtime());
  echo $s;
  sleep(5);
  $guarded_var->setVal($s);
  sleep(5);
  echo $guarded_var->getVal();
  $guarded_var->endSection();
?>
```

```
andrey@poohie:~/test/phpconf> php shm2.php
putVal() at : 0.99202900 1082906968
putVal() at : 0.99202900 1082906968
```

```
andrey@poohie:~/test/phpconf> php shm2.php
putVal() at : 0.99968900 1082906978
putVal() at : 0.99968900 1082906978
```

OO memory queue (0)

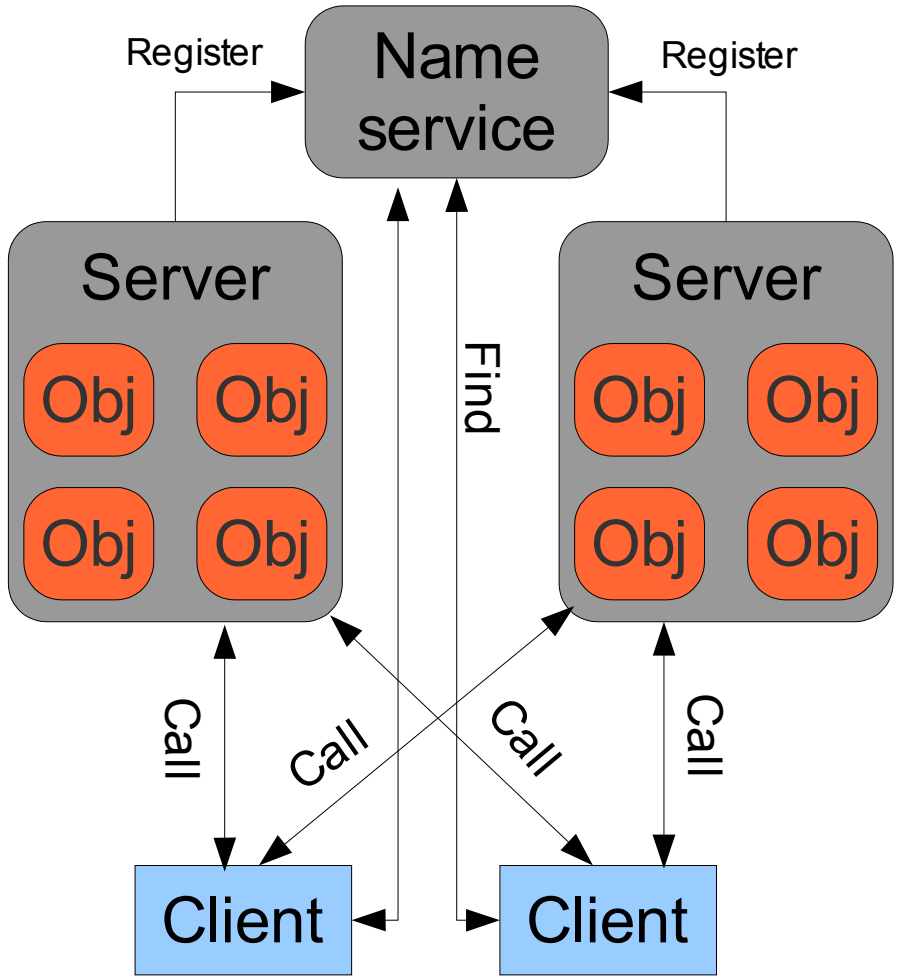
- Less details exposed to the programmer
- Still quite powerful since all things that can be done in a procedural way are possible by using the OO memory queue
- Less and more structured code which is easier to read and less error-prone.

OO memory queue - API

- ::__construct($shm_name,$size=16384,$perm='666')
- ::setReceiveOptions($options)
- ::setSendOptions($options)
- ::send($message, $options = array())
- ::receive($options = array())
- ::getRecvMsg()
- ::getRecvMsgType()
- ::getErrorCode()

Message based RPC

- MSGRPC is an example of using memory queues.
- There are 3 parts in the system :
  - Client
  - Server
  - Name Service (also a server)
- Steps (client):
  - Create an object – factory
  - Implicit lookup is made
  - Use the object
- Steps (server):
  - Create the server
  - Add servants
  - Register them



Use case : persistent data

- Good for cases when extracting data from different resources takes time or cache access time have to be low (no disk caching).
- Example : pre-parsed XML configuration file. The config file rarely changes -> put into shared memory on first script load.
- Get the configuration from the shared memory on every next request.
  Note : deserialization needs some time.
- If you cache strings then better use ext/shmop since it does not serialize the content.

Use case : XML example

```
<?php
require_once 'shm.php';
function rec($obj) {
  $ret_val = array();
  foreach ($obj as $k => $v) {
    if (($tmp = rec($v)) === NULL) {
      list(,$vv) = each($v);
      if (array_key_exists($k, $ret_val)) {
        $ret_val[$k][0] = $ret_val[$k];
      }
      if (is_array($ret_val[$k]))
        $ret_val[$k][] = $vv;
      else
        $ret_val[$k] = $vv;
    } else {
      if (array_key_exists($k, $ret_val)
        && !(is_array($ret_val[$k])
        && array_key_exists(0, $ret_val[$k]))){
          $ret_val[$k] = array($ret_val[$k]);
      }// if
      if (is_array($ret_val[$k]))
        $ret_val[$k][] = $tmp;
      else
        $ret_val[$k] = $tmp;
    }// if
  }// foreach
  if (!count($ret_val)) return NULL;
  return $ret_val;
}// rec

$shm = new Shm_Protected_Var("cff2", 16384,
           "666");
$shm->startSection();
if (!($configuration = $shm->getVal())) {
  echo "Not cached\n";
  $xml = simplexml_load_file('server.config');

  $configuration = rec($xml);
  $shm->setVal($configuration);
} else {
  echo "Cached\n";
}
$shm->endSection();

var_dump($configuration);

?>
```

Use case : XML example : outcome

- Caching in shared memory with semaphore locking was about ~4x faster
- Pre-caching and no locking at later stage gives more ~2x performance boost.
- Total speed-up can be up to 10x – this saves minutes/hours of CPU time on busy machines.

Questions ?

I am reachable at
andrey.hristov_100days_de or **andrey_php_net**

This presentation is available at :
http://andrey.hristov.com/projects/php_stuff/pres/

SHM/SEM classes:
http://andrey.hristov.com/projects/php_stuff/shm.tar.gz

memcached : http://www.danga.com/memcached/
pecl/memcache : http://pecl.php.net/package/memcache
msgrpc : http://andrey.hristov.com/projects/msgrpc/