
Think
ThinkInLAMP.com
PHPCon 2015
北京站

关于我

- Github: https://github.com/matyhtf

一个PHP Web程序的执行过程

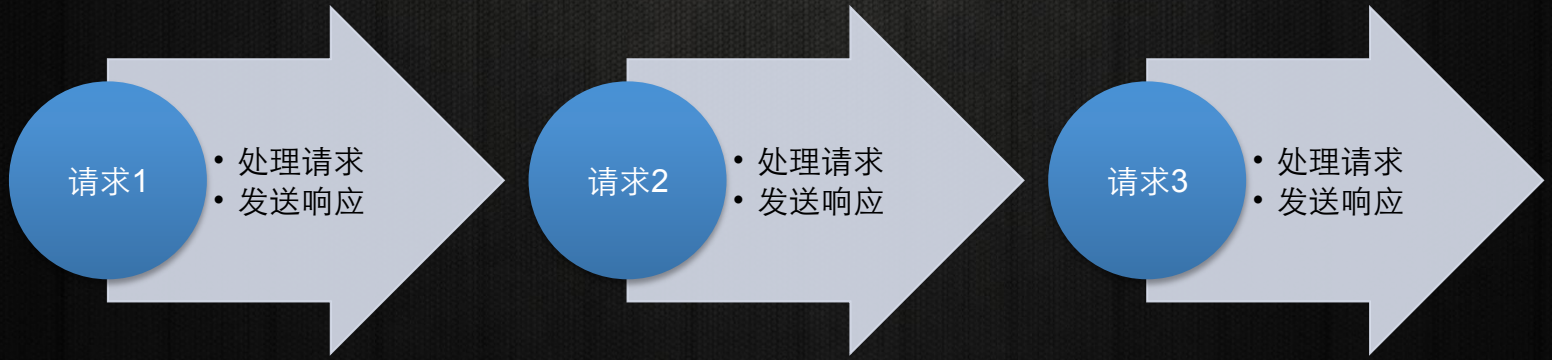
请求开始（Get/Post/Cookie/Session）

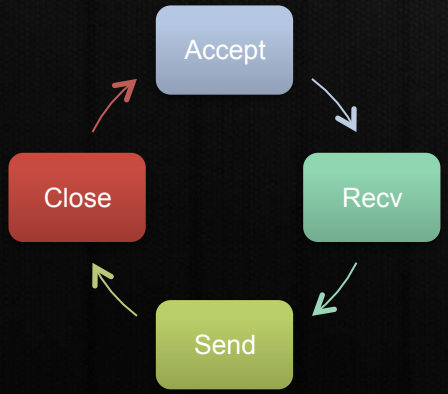
MySQL数据库查询/Redis查询

模板渲染输出HTML/json_encode

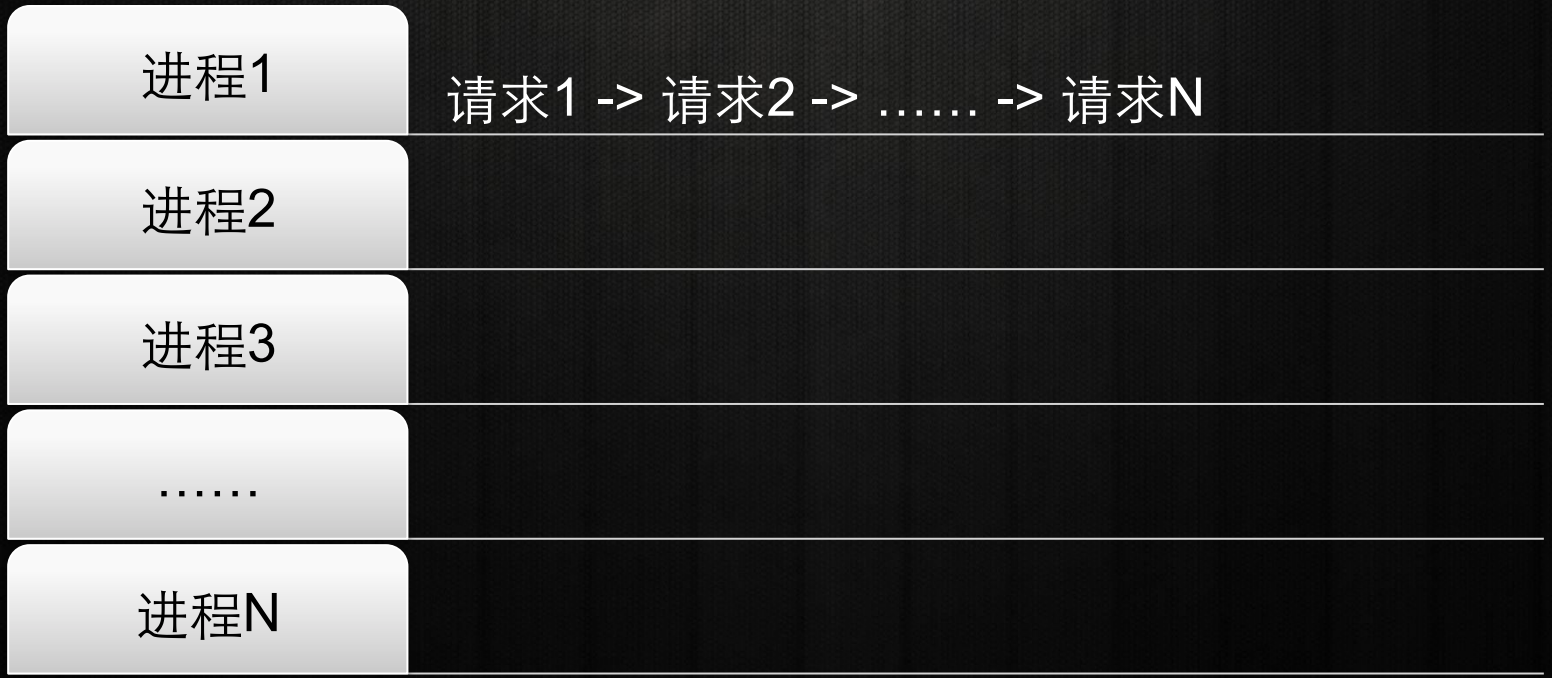
请求结束（回收所有内存和资源）

PHP-FPM进程的完整流程





多进程并发地处理请求



如何实现请求的公平分配?

leader follower_相关论文(共29210篇)_百度学术

...Nash equilibria, and multi-leader-follower games 《Computation... 被引频次: 42

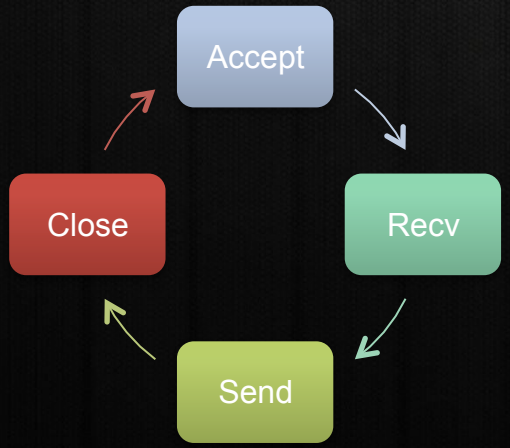
LEADER-FOLLOWER STRATEGIES FOR MULTILEVEL SYSTEMS 《Auto 被引频次: 35

matic Con...

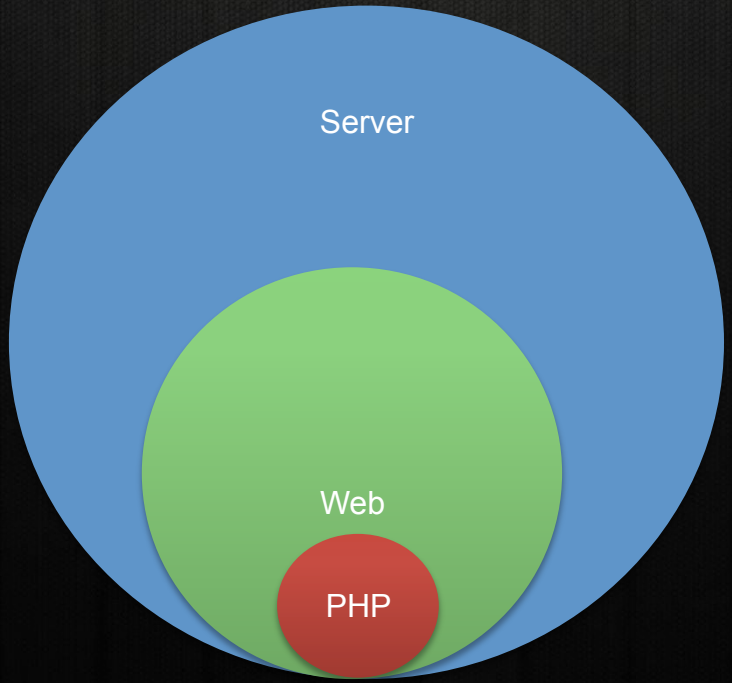
Vision-Based Localization for Leader–Follower ... 被引频次: 25

查看更多相关论文>>

xueshu.baidu.com



- Web服务也只是整个服务器端的一部分而已。Web之外还有FTP文件服务、SMTP邮件、聊天、PUSH消息等



开始探索之旅，学习底层网络通信

备注：PPT中的所有代码都可以在我的gist主页上找到

https://gist.github.com/matyhtf

大家用过这些扩展吗?

- stream
- sockets
- libevent/event
- pcntl/posix
- pthread
- sysvsem/sysvmsg
- shmop/sysvshm

第一个Server，阻塞+fork子进程

```php
<?php
$serv = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr)
    or die("create server failed");

while(1) {
    $conn = stream_socket_accept($serv);
    if (pcntl_fork() == 0 ) {
        $request = fread($conn);
        //do some thing
        //$response = “hello world”;
        fwrite($response);
        fclose($conn);
        exit(0);
    }
}
```

第二个Server，改良版

```php
<?php
$serv = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr)
    or die("create server failed");

for($i=0; $i < 32; $i ++) {
    if (pcntl_fork() == 0 ) {
        while(1) {
            $conn = stream_socket_accept($serv);
            if ($conn == false) continue;
            $request = fread($conn);//do some thing
            //$response = “hello world”;
            fwrite($response);
            fclose($conn);
        }
        exit(0);
    }
}
```

初次尝试异步(1)

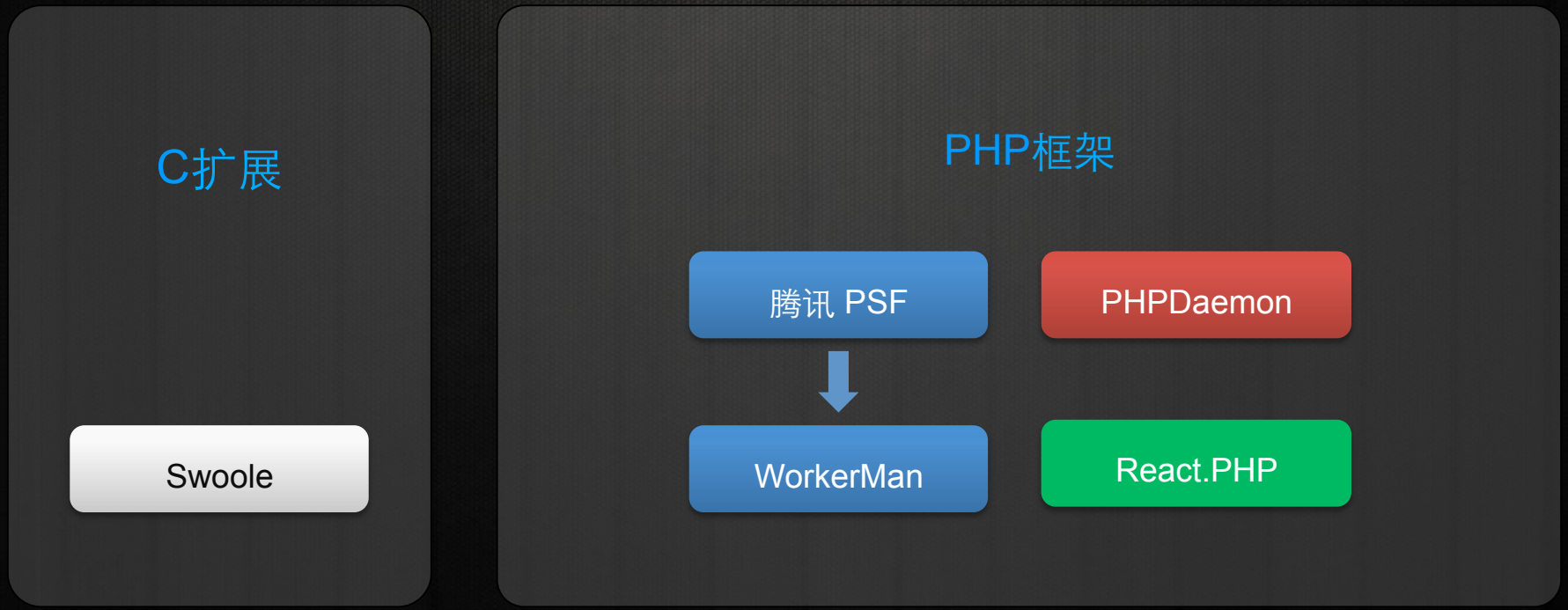
- 阅读圣经：《Unix网络编程》(UNP)
- select: stream_select / socket_select
- Nginx,memcache：libevent & epoll，异步的核心就是它
- PHP的libevent扩展、Event扩展

初次尝试异步(2)

```php
<?php
function read_cb($socket, $flag, $base) {
    fread($socket);
    fwrite("hello world\n");
}
function accept_cb($socket, $flag, $base) {
    $conn = stream_socket_accept($socket, 0);
    stream_set_blocking($conn, 0);
    $event = event_new();
    event_set($event, $conn, EV_READ | EV_PERSIST, 'read_cb'), $base);
    event_base_set($event, $base);
    event_add($event);
}
$serv = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr);
for($i=0; $i < 8; $i ++) {
    if (pcntl_fork() == 0) {
        $base = event_base_new();
        $event = event_new();
        event_set($event, $socket, EV_READ | EV_PERSIST, 'accept_cb', $base);
        event_base_set($event, $base);
        event_add($event);
        event_base_loop($base);
        exit(0);
    }
}
```

PHP实现Server的好处是什么？

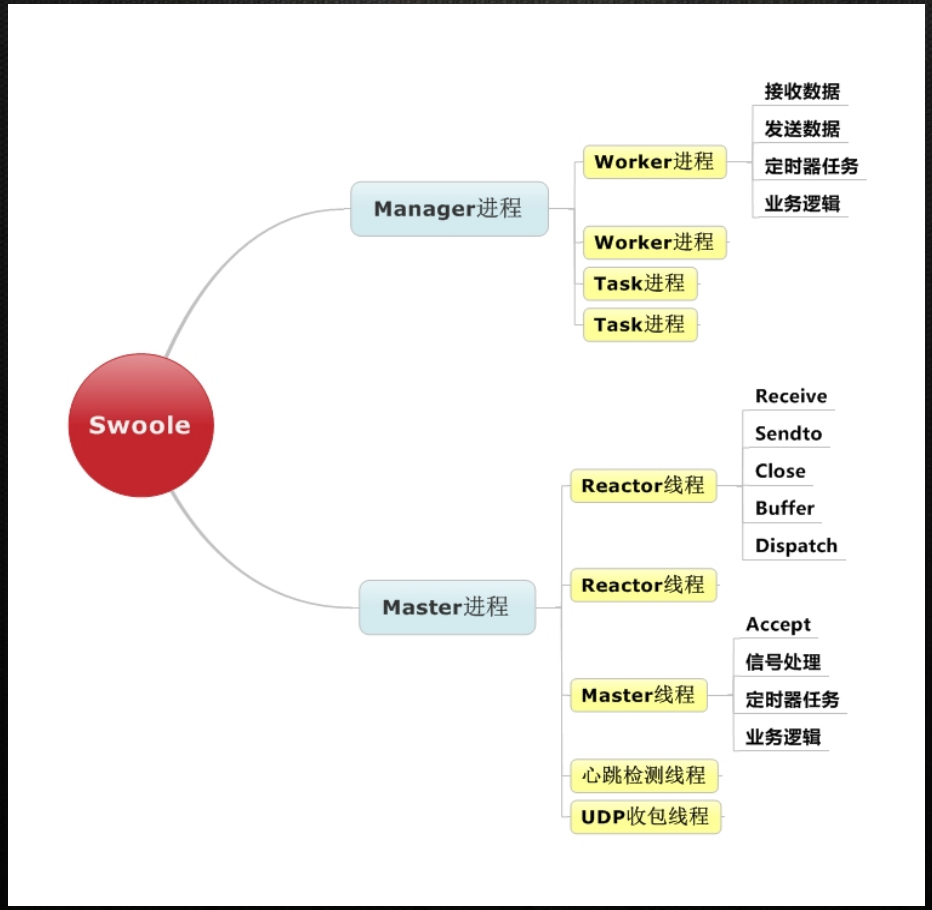
- 普通LAMP程序，PHP的所有对象请求时创建，请求结束时全部销毁。对于普通PHP程序来说避免了内存泄漏。对于大型网站来说，这是严重的资源浪费。
- PHP-Server中每次请求仅销毁与请求相关的对象。与请求无关的全局对象都不需要销毁。直接在下一次请求中复用。比如一个大数组，PHP-fpm每次都要构建HashTable，而PHP-Server程序完全不需要
- PHPer可以操控的范围更大了，局部缓存，Once操作，写文件合并，长连接，对象持久化，数据库连接池等都可以做。


C扩展
Swoole
PHP框架
腾讯 PSF
WorkerMan
PHPDaemon
React.PHP

为什么要用C扩展实现

- 性能，C代码实现的类和函数比PHP性能强10-100倍
- 内存，C扩展中对内存的控制可以精确到bit
- 数据结构，C扩展中可以针对不同的场景使用最优数据结构，而PHP代码中只有数组可用
- 直接操作底层，不需要依赖大量第三方扩展，而且粒度更小。如基于pcntl实现的信号处理，必须依赖PHP的tick机制，性能很差。
- 原子操作，C语言可以使用atomic操作实现自旋锁，原子自增/自减

开始使用Swoole扩展


Swoole
Manager进程
Worker进程
接收数据
发送数据
定时器任务
业务逻辑
Worker进程
Task进程
Task进程
Master进程
Reactor线程
Receive
Sendto
Close
Buffer
Dispatch
Reactor线程
Master线程
Accept
信号处理
定时器任务
业务逻辑
心跳检测线程
UDP收包线程

TCP-Server

```php
<?php
$serv = new swoole_server("127.0.0.1", 9501);

$serv->on('connect', function ($serv, $fd){
    echo "Client: Connected.\n";
});

$serv->on('receive', function ($serv, $fd, $from_id, $data){
    $serv->send($fd, 'Swoole: '.$data);
    $serv->close($fd);
});

$serv->on('close', function ($serv, $fd){
    echo "Client: Closed.\n";
});

$serv->start();
```

TCP-Async-Client

```php
$client = new swoole_client(SWOOLE_SOCK_TCP, SWOOLE_SOCK_ASYNC);

$client->on("connect", function($cli) {
    $cli->send("hello world\n");
});
$client->on("receive", function($cli, $data){
    echo "Received: ".$data."\n";
});
$client->on("error", function($cli){
    echo "Connect failed\n";
});
$client->on("close", function($cli){
    echo "Connection close\n";
});

$client->connect('127.0.0.1', 9501, 0.5);
```

TCP-Sync-Client

```php
$client = new swoole_client(SWOOLE_SOCK_TCP);
if (!$client->connect('127.0.0.1', 9501, 0.5))
{
    die("connect failed.");
}

if (!$client->send("hello world"))
{
    die("send failed.");
}

$data = $client->recv();
if (!$data)
{
    die("recv failed.");
}

$client->close();
```

异步MySQL

```php
$config = array(
    'host' => '127.0.0.1',
    'user' => 'root',
    'password' => 'root',
    'database' => 'test',
);
$pool = new Swoole\Async\MySQL($config, 100);
for($i = 0; $i < 10000; $i++)
{
    $pool->query("show tables", function($mysqli, mysqli_result $result){
        var_dump($result->fetch_all());
    });
}
```

异步Redis

```php
require __DIR__.'/src/Swoole/Async/RedisClient.php';
$redis = new Swoole\Async\RedisClient('127.0.0.1');

$redis->select('2', function () use ($redis) {
    $redis->set('key', 'value-rango', function ($result, $success) use ($redis) {
        for ($i = 0; $i < 3; $i++) {
            $redis->get('key', function ($result, $success) {
                echo "redis ok:\n";
                var_dump($success, $result);
            });
        }
    });
});
```

异步任务

```php
<?php
$server = new swoole_http_server('0.0.0.0', 8080);

$server->on('Task', function ($serv, $task_id, $from_id, $task) {
    var_dump($task['fd'], $task['get']);
    return "<h1> hello world </h1>";
});

$server->on('Finish', function ($serv, $task_id, $data){
    $resp = $server->_pool[$task_id];
    $resp->end($data);
});

$server->on('Request', function ($req, $resp) use ($server) {
    $task_id = $server->task(['fd' => $req->fd, 'get' => $req->get);
    $server->_pool[$task_id] = $resp;
});

$server->start();
```

Http服务器

```php
$http = new swoole_http_server("0.0.0.0", 9501);

$http->on('request', function ($request, $response) {
    $response->header("Content-Type", "text/html; charset=utf-8");
    $response->end("<h1>Hello Swoole. #".rand(1000, 9999)."</h1>");
});

$http->start();
```

WebSocket服务器

```php
$ws = new swoole_websocket_server("0.0.0.0", 9502);

$ws->on('open', function ($ws, $request) {
    var_dump($request->fd, $request->get, $request->server);
    $ws->push($request->fd, "hello, welcome\n");
});

$ws->on('message', function ($ws, $frame) {
    echo "Message: {$frame->data}\n";
    $ws->push($frame->fd, "server: {$frame->data}");
});

$ws->on('close', function ($ws, $fd) {
    echo "client-{$fd} is closed\n";
});

$ws->start();
```

使用场景

- swoole_http_server + redis-async + mysql-async + tcp-client- async，编写多进程全异步的Web程序
- TCP-server + TCP-Client 实现SOA服务器，php-fpm中使用TCP-Client + select实现并发请求。PHP实现4层架构、服务化治理
- swoole_websocket_server实现WebIM和PUSH系统
- 基于异步任务实现非响应逻辑的异步化，如在Http请求中发送邮件、发送短信

毫秒定时器

```php
//interval 2000ms
$serv->tick(2000, function ($timer_id) {
    echo "tick-2000ms\n";
});

//after 3000ms
$serv->after(3000, function () {
    echo "after 3000ms.\n"
});
```

异步Cli程序

```php
<?php
//从键盘输入数据
swoole_event_add(STDIN, function($stream) {
    $cmd = fgets($stream);
    echo $cmd;
});

//注册信号，优雅地退出
swoole_process::signal(SIGTERM, function () {
    echo "exit\n";
    swoole_event_exit();
});
```

并发HashTable

```php
<?php
$table = new swoole_table(1024);
$table->column('id', swoole_table::TYPE_INT, 4);
$table->column('name', swoole_table::TYPE_STRING, 64);
$table->column('num', swoole_table::TYPE_FLOAT);
$table->create();
//child
if (pcntl_fork() == 0) {
    $table->set('tianfenghan@qq.com', array(
        'id' => 145, 'name' => 'rango', 'num' => 3.1415));
}
//parent
else {
    sleep(1);
    var_dump($table->get('tianfenghan@qq.com'));
}
```

多进程/IPC/消息队列

```php
<?php
$process = new swoole_process(function($_process){
    $msg = $_process->read();
    echo "master: $msg";
    $_process->push($msg);
}, true);

$process->useQueue(ftok(__FILE__));
$process->start();

while (true) {
    $process->write("hello child\n");
    $msg = $_process->pop();
    echo "child: $msg"
    sleep(1);
}
```

Q & A