init.php

```php
<?php

/**
* Инициализируйте некоторые значения по умолчанию, необходимые для DokuWiki
*/

use dokuwiki\Extension\PluginController;
use dokuwiki\ErrorHandler;
use dokuwiki\Input\Input;
use dokuwiki\Extension\Event;
use dokuwiki\Extension\EventHandler;

/**
* время выполнения Dokuwiki
*
* @param   целое число $ начало
*
* @return   смешанный
*/
function delta_time($start = 0)
{
    return microtime(true) - ((float)$start);
}
define('DOKU_START_TIME', delta_time());

global $config_cascade;
$config_cascade = [];

// если доступно, загрузите файл конфигурации предварительной загрузки
$preload = fullpath(__DIR__) . '/preload.php';
if (file_exists($preload)) include($preload);

// определить путь включения
if (!defined('DOKU_INC')) define('DOKU_INC', fullpath(__DIR__ .
'/../') . '/');

// определить каталог плагина
if (!defined('DOKU_PLUGIN'))  define('DOKU_PLUGIN', DOKU_INC .
'lib/plugins/');

// определить путь конфигурации (упаковщики могут захотеть изменить его на /
etc / dokuwiki /)
if (!defined('DOKU_CONF')) define('DOKU_CONF', DOKU_INC . 'conf/');

// проверьте переопределение сообщений об ошибках или установите для
сообщений об ошибках разумные значения
if (!defined('DOKU_E_LEVEL') && file_exists(DOKU_CONF .
'report_e_all')) {
    define('DOKU_E_LEVEL', E_ALL);
```

```php
}
if (!defined('DOKU_E_LEVEL')) {
    error_reporting(E_ALL & ~E_NOTICE & ~E_DEPRECATED);
} else {
    error_reporting(DOKU_E_LEVEL);
}

// избегайте проблем с кэшированием #1594
header('Vary: Cookie');

// инициализация кэшей памяти
global $cache_revinfo;
       $cache_revinfo = [];
global $cache_wikifn;
       $cache_wikifn = [];
global $cache_cleanid;
       $cache_cleanid = [];
global $cache_authname;
       $cache_authname = [];
global $cache_metadata;
       $cache_metadata = [];

// всегда включайте ' inc / config_cascade.php '
// ранее заданные в preload.php поля $config_cascade будут объединены со значениями по умолчаниюs
include(DOKU_INC . 'inc/config_cascade.php');

//подготовить массив конфигурации()
global $conf;
$conf = [];

// загрузить глобальный файл(ы) конфигурации
foreach (['default', 'local', 'protected'] as $config_group) {
    if (empty($config_cascade['main'][$config_group])) continue;
    foreach ($config_cascade['main'][$config_group] as $config_file) {
        if (file_exists($config_file)) {
            include($config_file);
        }
    }
}

//подготовить массив лицензий()
global $license;
$license = [];

// загрузить файл(ы) лицензии
foreach (['default', 'local'] as $config_group) {
    if (empty($config_cascade['license'][$config_group])) continue;
    foreach ($config_cascade['license'][$config_group] as $config_file) {
```

```php
        if (file_exists($config_file)) {
            include($config_file);
        }
    }
}

// установить часовой пояс (как в днях до 5.3.0)
date_default_timezone_set(@date_default_timezone_get());

// определить baseURL
if (!defined('DOKU_REL')) define('DOKU_REL', getBaseURL(false));
if (!defined('DOKU_URL')) define('DOKU_URL', getBaseURL(true));
if (!defined('DOKU_BASE')) {
    if ($conf['canonical']) {
        define('DOKU_BASE', DOKU_URL);
    } else {
        define('DOKU_BASE', DOKU_REL);
    }
}

// определяем пробелы
if (!defined('NL')) define('NL', "\n");
if (!defined('DOKU_LF')) define('DOKU_LF', "\n");
if (!defined('DOKU_TAB')) define('DOKU_TAB', "\t");

// определить cookie и идентификатор сеанса, добавить порт сервера, если настроен securecookie FS#1664
if (!defined('DOKU_COOKIE')) {
    $serverPort = $_SERVER['SERVER_PORT'] ?? '';
    define('DOKU_COOKIE', 'DW' . md5(DOKU_REL . (($conf['securecookie']) ? $serverPort : '')));
    unset($serverPort);
}

// определить основной скрипт
if (!defined('DOKU_SCRIPT')) define('DOKU_SCRIPT', 'doku.php');

if (!defined('DOKU_TPL')) {
    /**
     * @deprecated 2012-10-13 заменен более динамичным методом
     * @see tpl_basedir()
     */
    define('DOKU_TPL', DOKU_BASE . 'lib/tpl/' . $conf['template'] . '/');
}

if (!defined('DOKU_TPLINC')) {
    /**
     * @deprecated 2012-10-13 replaced by more dynamic method
     * @see tpl_incdir()
     */
```

```
141     define('DOKU_TPLINC', DOKU_INC . 'lib/tpl/' . $conf['template']
. '/');
142 }
143
144 // сделать перезапись сеанса XHTML-совместимой
145 @ini_set('arg_separator.output', '&amp;');
146
147 // убедитесь, что глобальная zlib не мешает FS#1132
148 @ini_set('zlib.output_compression', 'off');
149
150 // увеличить предел возврата PCRE
151 @ini_set('pcre.backtrack_limit', '20971520');
152
153 // включить сжатие gzip, если поддерживается
154 $httpAcceptEncoding = $_SERVER['HTTP_ACCEPT_ENCODING'] ?? '';
155 $conf['gzip_output'] &= (strpos($httpAcceptEncoding, 'gzip') !==
false);
156 global $ACT;
157 if (
158     $conf['gzip_output'] &&
159         !defined('DOKU_DISABLE_GZIP_OUTPUT') &&
160         function_exists('ob_gzhandler') &&
161         // Disable compression when a (compressed) sitemap might be
delivered
162         // See
https://bugs.dokuwiki.org/index.php?do=details&task_id=2576
163         $ACT != 'sitemap'
164 ) {
165     ob_start('ob_gzhandler');
166 }
167
168 // инициализация сеанса
169 if (!headers_sent() && !defined('NOSESSION')) {
170     if (!defined('DOKU_SESSION_NAME'))
define('DOKU_SESSION_NAME', "DokuWiki");
171     if (!defined('DOKU_SESSION_LIFETIME'))
define('DOKU_SESSION_LIFETIME', 0);
172     if (!defined('DOKU_SESSION_PATH')) {
173         $cookieDir = empty($conf['cookiedir']) ? DOKU_REL :
$conf['cookiedir'];
174         define('DOKU_SESSION_PATH', $cookieDir);
175     }
176     if (!defined('DOKU_SESSION_DOMAIN'))
define('DOKU_SESSION_DOMAIN', '');
177
178     // начать сеанс
179     init_session();
180
181     // загрузить оставшиеся сообщения
182     if (isset($_SESSION[DOKU_COOKIE]['msg'])) {
183         $MSG = $_SESSION[DOKU_COOKIE]['msg'];
```

```php
        unset($_SESSION[DOKU_COOKIE]['msg']);
    }
}

// не позволяйте куки-файлам вмешиваться в переменные запроса
$_REQUEST = array_merge($_GET, $_POST);

// мы не хотим, чтобы URL-адрес очистки был выкопан
if (isset($_REQUEST['purge']) && !empty($_SERVER['HTTP_REFERER']))
unset($_REQUEST['purge']);

// предварительно рассчитать режимы создания файла
init_creationmodes();

// создаем реальные пути и проверяем их
init_paths();
init_files();

// настройка класса контроллера плагина (можно перезаписать в preload.php )
global $plugin_controller_class, $plugin_controller;
if (empty($plugin_controller_class)) $plugin_controller_class =
PluginController::class;

// автозагрузчик
require_once(DOKU_INC . 'inc/load.php');

// отныне все является исключением
ErrorHandler::register();

// отключаем gzip, если недоступно
define('DOKU_HAS_BZIP', function_exists('bzopen'));
define('DOKU_HAS_GZIP', function_exists('gzopen'));
if ($conf['compression'] == 'bz2' && !DOKU_HAS_BZIP) {
    $conf['compression'] = 'gz';
}
if ($conf['compression'] == 'gz' && !DOKU_HAS_GZIP) {
    $conf['compression'] = 0;
}

// класс обработки входных данных
global $INPUT;
$INPUT = new Input();

// инициализируем контроллер плагина
$plugin_controller = new $plugin_controller_class();

// инициализируем обработчик событий
global $EVENT_HANDLER;
$EVENT_HANDLER = new EventHandler();

$local = $conf['lang'];
```

```
233 Event::createAndTrigger('INIT_LANG_LOAD', $local, 'init_lang',
true);
234
235
236 // настройка системы аутентификации
237 if (!defined('NOSESSION')) {
238     auth_setup();
239 }
240
241 // настройка почтовой системы
242 mail_setup();
243
244 $nil = null;
245 Event::createAndTrigger('DOKUWIKI_INIT_DONE', $nil, null, false);
246
247 /**
248 * Инициализирует сеанс
249 *
250 * Проверяет, что переданный сеансовый  cookie-файл действителен,
недействительные игнорируются, и выдается новый идентификатор сеанса
251 *
252 * @ссылка  http://stackoverflow.com/a/33024310/172068
253 * @ссылка
http://php.net/manual/en/session.configuration.php#ini.session.sid-leng
th
254 */
255 function init_session()
256 {
257     global $conf;
258     session_name(DOKU_SESSION_NAME);
259     session_set_cookie_params([
260         'lifetime' => DOKU_SESSION_LIFETIME,
261         'path' => DOKU_SESSION_PATH,
262         'domain' => DOKU_SESSION_DOMAIN,
263         'secure' => ($conf['securecookie'] && is_ssl()),
264         'httponly' => true,
265         'samesite' => 'Lax',
266     ]);
267
268     // make sure the session cookie contains a valid session ID
269     if (isset($_COOKIE[DOKU_SESSION_NAME]) && !preg_match('/^[-,a-
zA-Z0-9]{22,256}$/', $_COOKIE[DOKU_SESSION_NAME])) {
270         unset($_COOKIE[DOKU_SESSION_NAME]);
271     }
272
273     session_start();
274 }
275
276
277 /**
278 * Проверяет пути из файла конфигурации
```

```php
 */
function init_paths()
{
    global $conf;

    $paths = [
        'datadir'   => 'pages',
        'olddir'    => 'attic',
        'mediadir'  => 'media',
        'mediaolddir' => 'media_attic',
        'metadir'   => 'meta',
        'mediametadir' => 'media_meta',
        'cachedir'  => 'cache',
        'indexdir'  => 'index',
        'lockdir'   => 'locks',
        'tmpdir'    => 'tmp',
        'logdir'    => 'log',
    ];

    foreach ($paths as $c => $p) {
        $path = empty($conf[$c]) ? $conf['savedir'] . '/' . $p :
$conf[$c];
        $conf[$c] = init_path($path);
        if (empty($conf[$c])) {
            $path = fullpath($path);
            nice_die("The $c ('$p') at $path is not found, isn't
accessible or writable.
                You should check your config and permission
settings.
                Or maybe you want to <a href=\"install.php\">run
the
                installer</a>?");
        }
    }

    // путь к старому списку изменений нужен только для обновления
    $conf['changelog_old'] = init_path(
        $conf['changelog'] ?? $conf['savedir'] . '/changes.log'
    );
    if ($conf['changelog_old'] == '') {
        unset($conf['changelog_old']);
    }
    // жестко запрограммирован журнал изменений, поскольку теперь это кэш,
который находится в метаданных
    $conf['changelog'] = $conf['metadir'] . '/_dokuwiki.changes';
    $conf['media_changelog'] = $conf['metadir'] .
'/_media.changes';
}

/**
 * Загрузить языковые строки
```

```
324  *
325  * @param string $langCode код языка, переданный обработчиком событий
326  */
327 function init_lang($langCode)
328 {
329     //подготовить языковой массив
330     global $lang, $config_cascade;
331     $lang = [];
332
333     //загрузить языковые файлы
334     require(DOKU_INC . 'inc/lang/en/lang.php');
335     foreach ($config_cascade['lang']['core'] as $config_file) {
336         if (file_exists($config_file . 'en/lang.php')) {
337             include($config_file . 'en/lang.php');
338         }
339     }
340
341     if ($langCode && $langCode != 'en') {
342         if (file_exists(DOKU_INC . "inc/lang/$langCode/lang.php"))
343 {
344             require(DOKU_INC . "inc/lang/$langCode/lang.php");
345         }
346         foreach ($config_cascade['lang']['core'] as $config_file) {
347             if (file_exists($config_file . "$langCode/lang.php")) {
348                 include($config_file . "$langCode/lang.php");
349             }
350         }
351     }
352
353 /**
354 * Проверяет наличие определенных файлов и создает их, если они отсутствуют.
355  */
356 function init_files()
357 {
358     global $conf;
359
360     $files = [$conf['indexdir'] . '/page.idx'];
361
362     foreach ($files as $file) {
363         if (!file_exists($file)) {
364             $fh = @fopen($file, 'a');
365             if ($fh) {
366                 fclose($fh);
367                 if ($conf['fperm']) chmod($file, $conf['fperm']);
368             } else {
369                 nice_die("$file is not writable. Check your
permissions settings!");
370             }
371         }
372     }
```

```php
}

/**
* Возвращает абсолютный путь
*
* Сначала проверяется указанный путь, затем DOKU_INC.
* Проверьте также доступность каталогов.
*
* @автор Андреас Гор <andi@splitbrain.org>
*
* @param  string $ path
*
* @return  bool | строка
*/
function init_path($path)
{
    // проверка существования
    $p = fullpath($path);
    if (!file_exists($p)) {
        $p = fullpath(DOKU_INC . $path);
        if (!file_exists($p)) {
            return '';
        }
    }

    // проверка возможности записи
    if (!@is_writable($p)) {
        return '';
    }

    // проверка доступности (бит выполнения) для каталогов
    if (@is_dir($p) && !file_exists("$p/.")) {
        return '';
    }

    return $p;
}

/**
* Устанавливает внутренние значения конфигурации fperm и dperm, которые, если установлены,
* будет использоваться для изменения разрешения вновь созданного каталога или
* файл с chmod. Учитывает влияние umask системы
* установка значений только при необходимости.
*/
function init_creationmodes()
{
    global $conf;

    // Устаревшая поддержка старой схемы umask / dmask
    unset($conf['dmask']);
```

```
423     unset($conf['fmask']);
424     unset($conf['umask']);
425
426     $conf['fperm'] = false;
427     $conf['dperm'] = false;
428
429     // получить системную маску umask, вернуться к 0, если она недоступна
430     $umask = @umask();
431     if (!$umask) $umask = 0000;
432
433     // проверка того, что автоматически устанавливается системой при создании
файлап
434     // и устанавливаем параметр fperm, если это не то, что нам нужно
435     $auto_fmode = 0666 & ~$umask;
436     if ($auto_fmode != $conf['fmode']) $conf['fperm'] =
$conf['fmode'];
437
438     // проверка того, что автоматически устанавливается системой при создании
каталога
439     // и устанавливаем параметр dperm, если это не то, что нам нужно.
440     $auto_dmode = 0777 & ~$umask;
441     if ($auto_dmode != $conf['dmode']) $conf['dperm'] =
$conf['dmode'];
442 }
443
444 /**
445 * Возвращает полный абсолютный URL-адрес каталога, где
446  * DokuWiki установлен в (включая завершающий слеш)
447 *
448 * !! Невозможно получить доступ к значениям $_SERVER через $INPUT
449 * !! здесь, поскольку эта функция вызывается до $INPUT
450 * !! инициализировано.
451 *
452 * @автор Андреас Гор <andi@splitbrain.org>
453 *
454  * @param  null | bool $ abs Вернуть абсолютный URL? (по умолчанию
null — $conf['canonical'])
455 *
456 * @возвращаемая строка
457 */
458 function getBaseURL($abs = null)
459 {
460     global $conf;
461
462     $abs ??= $conf['canonical'];
463
464     if (!empty($conf['basedir'])) {
465         $dir = $conf['basedir'];
466     } elseif (substr($_SERVER['SCRIPT_NAME'], -4) == '.php') {
467         $dir = dirname($_SERVER['SCRIPT_NAME']);
468     } elseif (substr($_SERVER['PHP_SELF'], -4) == '.php') {
```

```
469         $dir = dirname($_SERVER['PHP_SELF']);
470     } elseif ($_SERVER['DOCUMENT_ROOT'] &&
$_SERVER['SCRIPT_FILENAME']) {
471         $dir = preg_replace(
472             '/^' . preg_quote($_SERVER['DOCUMENT_ROOT'], '/') .
'/',
473             '',
474             $_SERVER['SCRIPT_FILENAME']
475         );
476         $dir = dirname('/' . $dir);
477     } else {
478         $dir = ''; //возможно, это неверно, но мы предполагаем, что он в
корне
479     }
480
481     $dir = str_replace('\\', '/', $dir);              // исправление
странного поведения WIN
482     $dir = preg_replace('#//+#', '/', "/$dir/");      // гарантируем
начальные и конечные слеши
483
484     //handle script in lib/exe dir
485     $dir = preg_replace('!lib/exe/$!', '', $dir);
486
487     //handle script in lib/plugins dir
488     $dir = preg_replace('!lib/plugins/.*$!', '', $dir);
489
490     //завершить здесь для относительных URL-адресов
491     if (!$abs) return $dir;
492
493     //используйте конфигурацию, если она доступна, обрежьте все слеши в конце
baseurl, чтобы избежать нескольких последовательных слешей в пути
494     if (!empty($conf['baseurl'])) return rtrim($conf['baseurl'],
'/') . $dir;
495
496     //разделить заголовок хоста на хост и порт
497     if (isset($_SERVER['HTTP_HOST'])) {
498         if (
499             (!empty($conf['trustedproxy'])) &&
isset($_SERVER['HTTP_X_FORWARDED_HOST'])
500              && preg_match('/' . $conf['trustedproxy'] . '/',
$_SERVER['REMOTE_ADDR'])
501         ) {
502             $cur_host = $_SERVER['HTTP_X_FORWARDED_HOST'];
503         } else {
504             $cur_host = $_SERVER['HTTP_HOST'];
505         }
506         $parsed_host = parse_url('http://' . $cur_host);
507         $host = $parsed_host['host'] ?? '';
508         $port = $parsed_host['port'] ?? '';
509     } elseif (isset($_SERVER['SERVER_NAME'])) {
510         $parsed_host = parse_url('http://' .
```

```
$_SERVER['SERVER_NAME']);
511         $host = $parsed_host['host'] ?? '';
512         $port = $parsed_host['port'] ?? '';
513     } else {
514         $host = php_uname('n');
515         $port = '';
516     }
517 
518     if (!is_ssl()) {
519         $proto = 'http://';
520         if ($port == '80') {
521             $port = '';
522         }
523     } else {
524         $proto = 'https://';
525         if ($port == '443') {
526             $port = '';
527         }
528     }
529 
530     if ($port !== '') $port = ':' . $port;
531 
532     return $proto . $host . $port . $dir;
533 }
534 
535 /**
536 * Проверьте, доступен ли сайт через  HTTPS
537 *
538 * Apache оставляет ,$_SERVER['HTTPS'] пустым, когда он недоступен, IIS
устанавливает его в значение «выкл.».
539 * «ложь» и «отключено» — это всего лишь предположения
540 *
541 * @returns bool true, когда SSL активен
542 */
543 function is_ssl()
544 {
545     global $conf;
546 
547     // проверяем, находимся ли мы за обратным прокси-сервером
548     if (
549         (!empty($conf['trustedproxy'])) &&
isset($_SERVER['HTTP_X_FORWARDED_PROTO'])
550          && preg_match('/' . $conf['trustedproxy'] . '/',
$_SERVER['REMOTE_ADDR'])
551          && ($_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https')
552     ) {
553         return true;
554     }
555 
556     if (preg_match('/^(|off|false|disabled)$/i', $_SERVER['HTTPS']
?? 'off')) {
```

```
557         return false;
558     }
559 
560     return true;
561 }
562 
563 /**
564  * проверяет, что это ОС Windows
565  * @return bool
566  */
567 function isWindows()
568 {
569     return strtoupper(substr(PHP_OS, 0, 3)) === 'WIN';
570 }
571 
572 /**
573  * вывести приятное сообщение, даже если ни один стили еще не загружен.
574  * @param  целое число | строка $ сообщение
575  * @param integer|string $msg
576  */
577 function nice_die($msg)
578 {
579     echo<<<EOT
580 <!DOCTYPE html>
581 <html>
582 <head><title>DokuWiki Setup Error</title></head>
583 <body style="font-family: Arial, sans-serif">
584     <div style="width:60%; margin: auto; background-color: #fcc;
585                 border: 1px solid #faa; padding: 0.5em 1em;">
586         <h1 style="font-size: 120%">DokuWiki Setup Error</h1>
587         <p>$msg</p>
588     </div>
589 </body>
590 </html>
591 EOT;
592     if (defined('DOKU_UNITTEST')) {
593         throw new RuntimeException('nice_die: ' . $msg);
594     }
595     exit(1);
596 }
597 
598 /**
599  * Замена realpath()
600  *
601  * Эта функция ведет себя аналогично функции realpath() в PHP, но не
разрешает
602  * символические ссылки или доступ к верхним каталогам
603  *
604  * @автор Андреас Гор <andi@splitbrain.org>
605  * @author <richpageau at yahoo dot co dot uk>
606  * @ссылка    http://php.net/manual/en/function.realpath.php#75992
```

```
607 *
608 * @param  string $ path
609 * @param  bool $ существует
610 *
611 * @return  bool | строка
612 */
613 function fullpath($path, $exists = false)
614 {
615     static $run = 0;
616     $root  = '';
617     $iswin = (isWindows() ||
!empty($GLOBALS['DOKU_UNITTEST_ASSUME_WINDOWS']));
618
619     // найти (неразрушаемый) корень пути - сохраняет содержимое  Windows
нетронутым
620     if ($path[0] == '/') {
621         $root = '/';
622     } elseif ($iswin) {
623         // сопоставить букву диска и пути  UNC
624         if (preg_match('!^([a-zA-z]:)(.*)!', $path, $match)) {
625             $root = $match[1] . '/';
626             $path = $match[2];
627         } elseif
(preg_match('!^(\\\\\\\\[^\\\\/]+\\\\[^\\\\/]+[\\\\/])(.*)!', $path,
$match)) {
628             $root = $match[1];
629             $path = $match[2];
630         }
631     }
632     $path = str_replace('\\', '/', $path);
633
634     // если указанный путь еще не был абсолютным, добавьте путь к скрипту и
повторите попытку
635     if (!$root) {
636         $base = dirname($_SERVER['SCRIPT_FILENAME']);
637         $path = $base . '/' . $path;
638         if ($run == 0) { // избегаем бесконечной рекурсии, когда база по
какой-то причине не является абсолютной
639             $run++;
640             return fullpath($path, $exists);
641         }
642     }
643     $run = 0;
644
645     // канонизировать
646     $path = explode('/', $path);
647     $newpath = [];
648     foreach ($path as $p) {
649         if ($p === '' || $p === '.') continue;
650         if ($p === '..') {
651             array_pop($newpath);
```

```
652                 continue;
653             }
654             $newpath[] = $p;
655         }
656         $finalpath = $root . implode('/', $newpath);
657 
658         // проверка существования при необходимости (кроме случаев модульного
тестирования)
659         if ($exists && !defined('DOKU_UNITTEST') &&
!file_exists($finalpath)) {
660             return false;
661         }
662         return $finalpath;
663 }
664 
```

«Подробности»

```
1<?php
2
3/**
4* Инициализируйте некоторые значения по умолчанию, необходимые для DokuWiki
5*/
6
7использовать  dokuwiki \ Extension \ PluginController ;
8использовать  dokuwiki \ ErrorHandler ;
9использовать  dokuwiki \ Ввод \ Ввод ;
10использовать  dokuwiki \ Extension \ Event ;
11использовать  dokuwiki \ Extension \ EventHandler ;
12
13/**
14* время выполнения Dokuwiki
15*
16* @param  целое число $ начало
17*
18* @return  смешанный
19*/
20функция  delta_time ($ start = 0 )
21{
22  вернуть микровремя ( true ) - (( float )$ start );
23}
24определить ( 'DOKU_START_TIME' , delta_time ());
25
26глобальный $ config_cascade ;
27$ config_cascade = [];
28
29// если доступно, загрузите файл конфигурации предварительной загрузки
30$ preload = полный_путь ( __DIR__ ). '/ preload.php ' ;
31если ( file_exists ($ preload )) включить ($ preload );
32
```

```
33// определить путь включения
34если (! определено ( 'DOKU_INC' )) определить ( 'DOKU_INC' , полный путь (
__DIR__ . '/../' ) . '/' );
35
36// определить каталог плагина
37если (! определено ( 'DOKU_PLUGIN' ))   определить ( 'DOKU_PLUGIN' , DOKU_INC
. ' lib / plugins /' );
38
39// определить путь конфигурации (упаковщики могут захотеть изменить его на / etc /
dokuwiki /)
40если (! определено ( 'DOKU_CONF' )) определить ( 'DOKU_CONF' , DOKU_INC .
'conf/' );
41
42// проверьте переопределение сообщений об ошибках или установите для сообщений об
ошибках разумные значения
43если (! определено ( 'DOKU_E_LEVEL' ) && file_exists ( DOKU_CONF .
'report_e_all' )) {
44  определить ( 'DOKU_E_LEVEL' , E_ALL );
45}
46если (! определено ( 'DOKU_E_LEVEL' )) {
47    error_reporting ( E_ALL & ~ E_NOTICE & ~ E_DEPRECATED );
48} еще {
49    error_reporting ( DOKU_E_LEVEL );
50}
51
52// избегайте проблем с кэшированием #1594
53заголовок ( 'Vary: Cookie' );
54
55// инициализация кэшей памяти
56глобальный $ cache_revinfo ;
57       $ cache_revinfo = [];
58глобальный $ cache_wikifn ;
59       $ cache_wikifn = [];
60глобальный $ cache_cleanid ;
61       $ cache_cleanid = [];
62глобальный $ cache_authname ;
63       $ cache_authname = [];
64глобальные $ cache_metadata ;
65       $ cache_metadata = [];
66
67// всегда включайте ' inc / config_cascade.php '
68// ранее заданные в preload.php поля $config_cascade будут объединены со
значениями по умолчанию
69включить ( DOKU_INC . ' inc / config_cascade.php ' );
70
71//подготовить массив конфигурации()
72глобальная $ conf ;
73$ conf = [];
74
75// загрузить глобальный файл(ы) конфигурации
76foreach ([ 'default' , 'local' , 'protected' ] как $ config_group ) {
```

```
77  если ( пусто ($ config_cascade [ 'main' ][$ config_group ])) продолжить ;
78    foreach ($ config_cascade [ 'main' ][$ config_group ] как $
config_file ) {
79    если ( file_exists ($ config_file )) {
80      включить ($ config_file );
81        }
82    }
83}
84
85//подготовить массив лицензий()
86глобальная лицензия $ ;
87$ лицензия = [];
88
89// загрузить файл(ы) лицензии
90foreach ([ 'default' , 'local' ] как $ config_group ) {
91  если ( пусто ($ config_cascade [ 'license' ][$ config_group ])) продолжить
;
92    foreach ($ config_cascade [ 'license' ][$ config_group ] как $
config_file ) {
93    если ( file_exists ($ config_file )) {
94      включить ($ config_file );
95        }
96    }
97}
98
99// установить часовой пояс (как в днях до 5.3.0)
100date_default_timezone_set (@date_default_timezone_get ( ));
101
102// определить baseURL
103if (! определено ( 'DOKU_REL' )) define ( 'DOKU_REL' , getBaseURL ( false
));
104если (! определено ( 'DOKU_URL' )) определить ( 'DOKU_URL' , getBaseURL (
true ));
105если (! определено ( 'DOKU_BASE' )) {
106  если ($ conf [ 'канонический']) {
107    определить ( 'DOKU_BASE' , DOKU_URL );
108  } еще {
109    определить ( 'DOKU_BASE' , DOKU_REL );
110    }
111}
112
113// определяем пробелы
114если (! определено ( 'NL' )) определить ( 'NL' , " \n " );
115если (! определено ( 'DOKU_LF' )) определить ( 'DOKU_LF' , " \n " );
116если (! определено ( 'DOKU_TAB' )) определить ( 'DOKU_TAB' , " \t " );
117
118// определить cookie и идентификатор сеанса, добавить порт сервера, если настроен
securecookie FS#1664
119если (! определено ( 'DOKU_COOKIE' )) {
120    $ serverPort = $ _SERVER [ 'SERVER_PORT' ] ?? '' ;
121    define ( 'DOKU_COOKIE' , 'DW' . md5 ( DOKU_REL . (($ conf [
```

```
'securecookie' ])? $ serverPort : '' )));
122  не установлен ($ serverPort );
123}
124
125// определить основной скрипт
126если (! определено ( 'DOKU_SCRIPT' )) определить ( 'DOKU_SCRIPT' , '
doku.php ' );
127
128если (! определено ( 'DOKU_TPL' )) {
129    /**
130     * @deprecated 2012-10-13 заменен более динамичным методом
131     * @see  tpl_basedir ()
132     */
133  определить ( 'DOKU_TPL' , DOKU_BASE . ' lib / tpl /' . $ conf [
'template' ] . '/' );
134}
135
136если (! определено ( 'DOKU_TPLINC' )) {
137    /**
138     * @deprecated 2012-10-13 заменен более динамичным методом
139     * @see  tpl_incdir ()
140     */
141  определить ( 'DOKU_TPLINC' , DOKU_INC . ' lib / tpl /' . $ conf [
'template' ] . '/' );
142}
143
144// сделать перезапись сеанса XHTML-совместимой
145@ ini_set ( 'arg_separator.output' , '&' );
146
147// убедитесь, что глобальная zlib не мешает FS#1132
148@ ini_set ( 'zlib.output_compression' , 'off' );
149
150// увеличить предел возврата PCRE
151@ ini_set ( 'pcre.backtrack_limit' , '20971520' );
152
153// включить сжатие gzip, если поддерживается
154$ httpAcceptEncoding = $ _SERVER [ 'HTTP_ACCEPT_ENCODING' ] ?? '' ;
155$ conf [ 'gzip_output' ] &= ( strpos ( $ httpAcceptEncoding , 'gzip' )
!== false );
156глобальный $ ACT ;
157если (
158    $ conf [ 'gzip_output' ] &&
159    ! определено ( 'DOKU_DISABLE_GZIP_OUTPUT' ) &&
160        function_exists ( 'ob_gzhandler' ) &&
161    // Отключить сжатие, если может быть доставлена (сжатая) карта сайта
162    // См. https://bugs.dokuwiki.org/index.php?do=details&task_id=2576
163        $ ACT != 'карта сайта'
164) {
165    ob_start ( 'ob_gzhandler' );
166}
167
```

```
168// инициализация сеанса
169если (! headers_sent () && ! defined ( 'NOSESSION' )) {
170    if (! определено ( 'DOKU_SESSION_NAME' ))      define (
'DOKU_SESSION_NAME' , "DokuWiki" );
171  если (! определено ( 'DOKU_SESSION_LIFETIME' )) определить (
'DOKU_SESSION_LIFETIME' , 0 );
172  если (! определено ( 'DOKU_SESSION_PATH' )) {
173        $ cookieDir = пусто ($ conf [ 'cookiedir' ]) ? DOKU_REL : $ conf
[ 'cookiedir' ];
174    определить ( 'DOKU_SESSION_PATH' , $ cookieDir );
175    }
176  если (! определено ( 'DOKU_SESSION_DOMAIN' ))    определить (
'DOKU_SESSION_DOMAIN' , '' );
177
178  // начать сеанс
179    init_session ();
180
181  // загрузить оставшиеся сообщения
182  если ( isset ($ _SESSION [ DOKU_COOKIE ][ 'сообщение' ])) {
183        $ MSG = $ _SESSION [ DOKU_COOKIE ][ 'сообщение' ];
184    сброшено ($ _SESSION [ DOKU_COOKIE ][ 'сообщение' ]);
185    }
186}
187
188// не позволяйте куки-файлам вмешиваться в переменные запроса
189$ _REQUEST = array_merge ($ _GET , $ _POST );
190
191// мы не хотим, чтобы URL-адрес очистки был выкопан
192если ( isset ($ _REQUEST [ 'purge' ]) && ! empty ($ _SERVER [
'HTTP_REFERER' ])) unset ($ _REQUEST [ 'purge' ]);
193
194// предварительно рассчитать режимы создания файла
195init_creationmodes ();
196
197// создаем реальные пути и проверяем их
198init_paths ();
199init_files ();
200
201// настройка класса контроллера плагина (можно перезаписать в preload.php )
202глобальный $ plugin_controller_class , $ plugin_controller ;
203если ( пусто ($ plugin_controller_class )) $ plugin_controller_class =
PluginController :: class ;
204
205// автозагрузчик
206require_once ( DOKU_INC . ' inc / load.php ' );
207
208// отныне все является исключением
209ErrorHandler :: register ();
210
211// отключаем gzip, если недоступно
212определить ( 'DOKU_HAS_BZIP' , function_exists ( 'bzopen' ));
```

```
213определить ( 'DOKU_HAS_GZIP' , function_exists ( 'gzopen' ));
214если ($ conf [ 'сжатие' ] == 'bz2' && ! DOKU_HAS_BZIP ) {
215    $ conf [ 'сжатие' ] = 'gz' ;
216}
217если ($ conf [ 'сжатие' ] == 'gz' && ! DOKU_HAS_GZIP ) {
218    $ conf [ 'сжатие' ] = 0 ;
219}
220
221// класс обработки входных данных
222глобальный $ ВХОД ;
223$ INPUT = новый Вход ();
224
225// инициализируем контроллер плагина
226$ plugin_controller = новый $ plugin_controller_class ();
227
228// инициализируем обработчик событий
229глобальный $ EVENT_HANDLER ;
230$ EVENT_HANDLER = новый  EventHandler ();
231
232$ local = $ conf [ 'язык' ];
233Событие :: createAndTrigger ( 'INIT_LANG_LOAD' , $ local , 'init_lang' ,
true );
234
235
236// настройка системы аутентификации
237если (! определено ( 'NOSESSION' )) {
238    auth_setup ();
239}
240
241// настройка почтовой системы
242mail_setup ();
243
244$ nil = ноль ;
245Событие :: createAndTrigger ( 'DOKUWIKI_INIT_DONE' , $ nil , null , false
);
246
247/**
248* Инициализирует сеанс
249*
250* Проверяет, что переданный сеансовый cookie-файл действителен, недействительные
игнорируются, и выдается новый идентификатор сеанса
251*
252* @ссылка  http://stackoverflow.com/a/33024310/172068
253* @ссылка
http://php.net/manual/en/session.configuration.php#ini.session.sid-length
254*/
255функция  init_session ()
256{
257  глобальная $ conf ;
258  имя_сеанса ( DOKU_SESSION_NAME );
259    session_set_cookie_params ([
```

```
260    'время жизни' => DOKU_SESSION_LIFETIME ,
261    'путь' => DOKU_SESSION_PATH ,
262    'домен' => DOKU_SESSION_DOMAIN ,
263        'secure' => ($ conf [ 'securecookie' ] && is_ssl ()),
264        'httponly' => правда ,
265        'samesite' => 'Лакс',
266    ]);
267
268  // убедитесь, что cookie-файл сеанса содержит действительный идентификатор
сеанса
269  если ( isset ($ _COOKIE [ ИМЯ_СЕАНСА_ДОКУ ]) && ! preg_match ( '/^[-,a-
zA-Z0-9]{22,256}$/' , $ _COOKIE [ ИМЯ_СЕАНСА_ДОКУ ])) {
270    не установлено ($ _COOKIE [ DOKU_SESSION_NAME ]);
271    }
272
273    session_start ();
274}
275
276
277/**
278* Проверяет пути из файла конфигурации
279*/
280функция  init_paths ()
281{
282  глобальная $ conf ;
283
284  $ пути = [
285        'datadir'    => 'страницы',
286        'olddir'     => 'чердак',
287        'mediadir'   => 'медиа',
288        'mediaolddir' => 'media_attic' ,
289        'metadir'    => 'мета',
290        'mediametadir' => 'media_meta' ,
291        'cachedir'   => 'кэш',
292        'indexdir'   => 'index' ,
293        'lockdir'    => 'замки',
294        'tmpdir'     => 'tmp' ,
295        'logdir'     => 'журнал',
296    ];
297
298    foreach ($ пути как $ c => $ p ) {
299        $ path = empty ($ conf [$ c ]) ? $ conf [ 'savedir' ] . '/' . $ p
: $ conf [$ c ];
300        $ conf [$ c ] = init_path ($ path );
301    если ( пусто ($ conf [$ c ])) {
302            $ path = полный_путь ($ path );
303            nice_die ( " $ c (' $ p ') по адресу $ path не найден,
недоступен или недоступен для записи.
304        Вам следует проверить конфигурацию и настройки разрешений.
305        Или, может быть, вы хотите <a href= \" install.php \" > запустить
306        установщик</a>?" );
```

```
307         }
308     }
309
310   // путь к старому списку изменений нужен только для обновления
311     $ conf [ 'changelog_old' ] = init_path (
312         $ conf [ 'changelog' ] ?? $ conf [ 'savedir' ] . '/changes.log'
313     );
314   если ($ conf [ 'changelog_old' ] == '' ) {
315     сброшено ($ conf [ 'changelog_old' ]);
316     }
317   // жестко запрограммирован журнал изменений, поскольку теперь это кэш, который
находится в метаданных
318     $ conf [ 'changelog' ] = $ conf [ 'metadir' ] . '/_dokuwiki.changes'
;
319     $ conf [ 'media_changelog' ] = $ conf [ 'metadir' ] .
'/_media.changes' ;
320}
321
322/**
323* Загрузить языковые строки
324*
325* @param  string $ langCode  код языка, переданный обработчиком событий
326*/
327функция  init_lang ($ langCode )
328{
329   //подготовить языковой массив
330   глобальный $ lang , $ config_cascade ;
331     $ lang = [];
332
333   //загрузить языковые файлы
334   требуется ( DOKU_INC . ' inc / lang / en / lang.php ' );
335     foreach ($ config_cascade [ 'lang' ][ 'core' ] как $ config_file ) {
336     если ( file_exists ($ config_file . ' en / lang.php ' )) {
337             include ( $ config_file.'en / lang.php ' ) ;
338         }
339     }
340
341     if ($ langCode && $ langCode != 'en' ) {
342         if ( file_exists ( DOKU_INC . " inc / lang / $ langCode /
lang.php " ) {
343       требуется ( DOKU_INC . " inc / lang / $ langCode / lang.php " );
344         }
345         foreach ($ config_cascade [ 'lang' ][ 'core' ] как $ config_file
) {
346       если ( file_exists ($ config_file . " $ langCode / lang.php " )) {
347                 include ($ config_file . " $ langCode / lang.php " );
348             }
349         }
350     }
351}
352
```

```
353/**
354* Проверяет наличие определенных файлов и создает их, если они отсутствуют.
355*/
356функция  init_files ()
357{
358  глобальная $ conf ;
359
360    $ files = [$ conf [ 'indexdir' ] . '/page.idx' ];
361
362    foreach ($ файлы как $ файл ) {
363    если (! file_exists ($ file )) {
364            $ fh = @ fopen ($ file , 'a' );
365      если ($ фх ) {
366                fclose ($ fh );
367        если ($ conf [ 'fperm' ]) chmod ($ file , $ conf [ 'fperm' ]);
368      } еще {
369                nice_die ( " $ file недоступен для записи. Проверьте настройки прав доступа!" );
370            }
371        }
372    }
373}
374
375/**
376* Возвращает абсолютный путь
377*
378* Сначала проверяется указанный путь, затем DOKU_INC.
379* Проверьте также доступность каталогов.
380*
381* @автор Андреас Гор <andi@splitbrain.org>
382*
383* @param  string $ path
384*
385* @return  bool | строка
386*/
387функция  init_path ($ path )
388{
389  // проверка существования
390    $ p = полный путь ($ path );
391  если (! file_exists ($ p )) {
392        $ p = полный_путь ( DOKU_INC . $ path );
393    если (! file_exists ($ p )) {
394      возвращаться '';
395        }
396    }
397
398  // проверка возможности записи
399  если (!@ is_writable ($ p )) {
400    возвращаться '';
401    }
402
```

```
403  // проверка доступности (бит выполнения) для каталогов
404  если (@is_dir ( $ p ) &&! file_exists ( " $ p /." )) {
405    возвращаться '';
406    }
407
408  вернуть $ p ;
409}
410
411/**
412* Устанавливает внутренние значения конфигурации fperm и dperm, которые, если
установлены,
413* будет использоваться для изменения разрешения вновь созданного каталога или
414* файл с chmod. Учитывает влияние umask системы
415* установка значений только при необходимости.
416*/
417функция  init_creationmodes ()
418{
419  глобальная $ conf ;
420
421  // Устаревшая поддержка старой схемы umask / dmask
422  сброшено ($ conf [ 'dmask' ]);
423  сброшено ($ conf [ 'fmask' ]);
424  не установлено ($ conf [ 'umask' ]);
425
426    $ conf [ 'fperm' ] = false ;
427    $ conf [ 'dperm' ] = false ;
428
429  // получить системную маску umask, вернуться к 0, если она недоступна
430    $ umask = @ umask ();
431  если (!$ umask ) $ umask = 0000 ;
432
433  // проверка того, что автоматически устанавливается системой при создании файла
434  // и устанавливаем параметр fperm, если это не то, что нам нужно
435    $ auto_fmode = 0666 & ~$ umask ;
436  если ($ auto_fmode != $ conf [ 'fmode' ]) $ conf [ 'fperm' ] = $ conf [
'fmode' ];
437
438  // проверка того, что автоматически устанавливается системой при создании каталога
439  // и устанавливаем параметр dperm, если это не то, что нам нужно.
440    $ auto_dmode = 0777 & ~$ umask ;
441  если ($ auto_dmode != $ conf [ 'dmode' ]) $ conf [ 'dperm' ] = $ conf [
'dmode' ];
442}
443
444/**
445* Возвращает полный абсолютный URL-адрес каталога, где
446* DokuWiki установлен в (включая завершающий слеш)
447*
448* !! Невозможно получить доступ к значениям $_SERVER через $INPUT
449* !! здесь, поскольку эта функция вызывается до $INPUT
450* !! инициализировано.
```

```
451*
452* @автор Андреас Гор <andi@splitbrain.org>
453*
454* @param  null | bool $ abs Вернуть абсолютный URL? (по умолчанию null — $conf['canonical'])
455*
456* @возвращаемая строка
457*/
458функция  getBaseURL ($ abs = null )
459{
460  глобальная $ conf ;
461
462    $ abs ??= $ conf [ 'канонический'];
463
464    if (! пусто ($ conf [ 'basedir' ])) {
465        $ dir = $ conf [ 'basedir' ];
466    } elseif ( substr ($ _SERVER [ 'SCRIPT_NAME' ], -4 ) == '.php' ) {
467        $ dir = dirname ($ _SERVER [ 'SCRIPT_NAME' ]);
468    } elseif ( substr ($ _SERVER [ 'PHP_SELF' ], -4 ) == '.php' ) {
469        $ dir = имя_каталога ($ _SERVER [ 'PHP_SELF' ]);
470    } elseif ($ _SERVER [ 'DOCUMENT_ROOT' ] && $ _SERVER [ 'SCRIPT_FILENAME' ]) {
471        $ dir = preg_replace (
472            '/^' . preg_quote ($ _SERVER [ 'DOCUMENT_ROOT' ], '/' ) . '/' ,
473            '' ,
474            $ _SERVER [ 'имя_файла_сценария']
475        );
476        $ dir = имя_каталога ( '/' . $ dir );
477  } еще {
478        $ dir = '' ; //возможно, это неверно, но мы предполагаем, что он в корне
479    }
480
481    $ dir = str_replace ( ' \\ ' , '/' , $ dir );              // исправление странного поведения  WIN
482    $ dir = preg_replace ( '#//+#' , '/' , "/ $ dir /" );      // гарантируем начальные и конечные слеши
483
484  //обработка скрипта в каталоге  lib / exe
485    $ dir = preg_replace ( '! lib / exe /$!' , '' , $ dir );
486
487  //обработка скрипта в каталоге  lib / plugins
488    $ dir = preg_replace ( '! lib / plugins /.*$!' , '' , $ dir );
489
490  //завершить здесь для относительных  URL-адресов
491  если (!$ abs ) вернуть $ dir ;
492
493  //используйте конфигурацию, если она доступна, обрежьте все слеши в конце baseurl,  чтобы избежать нескольких последовательных слешей в пути
494  если (! пусто ($ conf [ 'baseurl' ])) вернуть  rtrim ($ conf [ 'baseurl' ], '/' ) . $ dir ;
```

```
495
496  //разделить заголовок хоста на хост и порт
497  если ( isset ($ _SERVER [ 'HTTP_HOST' ])) {
498    если (
499      (! пусто ($ conf [ 'trustedproxy' ])) && isset ($ _SERVER [
'HTTP_X_FORWARDED_HOST' ])
500              && preg_match ( '/' . $ conf [ 'trustedproxy' ]. '/' , $
_SERVER [ 'REMOTE_ADDR' ])
501        ) {
502            $ cur_host = $ _SERVER [ 'HTTP_X_FORWARDED_HOST' ];
503    } еще {
504            $ cur_host = $ _SERVER [ 'HTTP_HOST' ];
505        }
506        $ parsed_host = parse_url ( ' http://' . $cur_host);
507        $host = $parsed_host[' хост '] ?? '';
508        $port = $parsed_host[' порт '] ?? '';
509    } elseif (isset($_SERVER[' ИМЯ_СЕРВЕРА ' ])) {
510        $parsed_host = parse_url(' http : //' . $_SERVER['SERVER_NAME']);
511        $ host = $ parsed_host [ 'host' ] ?? '' ;
512        $ port = $ parsed_host [ 'port' ] ?? '' ;
513  } еще {
514        $ host = php_uname ( 'n' );
515    $ порт = '' ;
516    }
517
518  если (! is_ssl ()) {
519        $ proto = ' http://';
520    если ($port == ' 80 ') {
521      $порт = '';
522        }
523  } еще {
524        $proto = ' https : //';
525    если ($ порт == '443' ) {
526      $ порт = '' ;
527        }
528    }
529
530  если ($ порт !== '' ) $ порт = ':' . $ порт ;
531
532    return $ proto .$ host .$ port .$ dir ;
533}
534
535/**
536* Проверьте, доступен ли сайт через  HTTPS
537*
538* Apache оставляет ,$_SERVER['HTTPS'] пустым, когда он недоступен, IIS
устанавливает его в значение «выкл.».
539* «ложь» и «отключено» — это всего лишь предположения
540*
541* @returns bool true, когда SSL активен
542*/
```

```
543функция  is_ssl ()
544{
545  глобальная $ conf ;
546
547  // проверяем, находимся ли мы за обратным прокси-сервером
548  если (
549    (! пусто ($ conf [ 'trustedproxy' ])) && isset ($ _SERVER [
'HTTP_X_FORWARDED_PROTO' ])
550         && preg_match ( '/' . $ conf [ 'trustedproxy' ]. '/' , $ _SERVER
[ 'REMOTE_ADDR' ])
551         && ($ _SERVER [ 'HTTP_X_FORWARDED_PROTO' ] == 'https' )
552    ) {
553    вернуть истину ;
554    }
555
556  если ( preg_match ( '/^(|выкл|ложь|отключено)$/i' , $ _SERVER [ 'HTTPS' ]
?? 'выкл' )) {
557    вернуть ложь ;
558    }
559
560  вернуть истину ;
561}
562
563/**
564* проверяет, что это ОС Windows
565* @return  bool
566*/
567функция  isWindows ()
568{
569    return  strtoupper ( substr ( PHP_OS , 0 , 3 )) === 'WIN' ;
570}
571
572/**
573* вывести приятное сообщение, даже если ни один стили еще не загружен.
574*
575* @param   целое число | строка $ сообщение
576*/
577функция  nice_die ($ msg )
578{
579  эхо <<< EOT
580<!DOCTYPE html>
581<html>
582<head><title>Ошибка установки DokuWiki</title></head>
583<body style="font-family: Arial, sans-serif">
584    <div style="width:60%; margin: auto; background-color: #fcc;
585        рамка: 1px сплошная #faa; отступ: 0.5em 1em;">
586        <h1 style="font-size: 120%">Ошибка установки DokuWiki</h1>
587        <p> $ сообщение </p>
588    </div>
589</тело>
590</html>
```

```
591CPB ;
592  если ( определено  ( 'DOKU_UNITTEST' )) {
593         throw  new  RuntimeException ( 'nice_die: ' . $ msg );
594     }
595  выход ( 1 );
596}
597
598/**
599* Замена realpath()
600*
601* Эта функция ведет себя аналогично функции realpath() в PHP, но не разрешает
602* символические ссылки или доступ к верхним каталогам
603*
604* @автор Андреас Гор <andi@splitbrain.org>
605* @author <richpageau at yahoo dot co dot uk>
606* @ссылка    http://php.net/manual/en/function.realpath.php#75992
607*
608* @param  string $ path
609* @param  bool $ существует
610*
611* @return  bool | строка
612*/
613функция  полный_путь ($ путь , $ существует = ложь )
614{
615  статический $ run = 0 ;
616  $ корень  = '' ;
617     $ iswin = ( isWindows () || ! пусто ($ GLOBALS [
'DOKU_UNITTEST_ASSUME_WINDOWS' ]));
618
619  // найти (неразрушаемый) корень пути - сохраняет содержимое Windows нетронутым
620  если ($ путь [ 0 ] == '/' ) {
621     $ корень = '/' ;
622     } elseif ($ iswin ) {
623     // сопоставить букву диска и пути UNC
624     если ( preg_match ( '!^([a-zA-z]:)(.*)!' , $ path , $ match )) {
625             $ root = $ match [ 1 ] . '/' ;
626       $ путь = $ совпадение [ 2 ];
627        } elseif ( preg_match ( '!^( \\ \\ \\ \\ [^ \\ \\ /]+ \\ \\ [^ \\
\\ /]+[ \\ \\ /])(.*)!' , $ path , $ match )) {
628       $ корень = $ матч [ 1 ];
629       $ путь = $ совпадение [ 2 ];
630        }
631     }
632     $ path = str_replace ( ' \\ ' , '/' , $ path );
633
634  // если указанный путь еще не был абсолютным, добавьте путь к скрипту и повторите
попытку
635  если (!$ корень ) {
636        $ base = dirname ($ _SERVER [ 'SCRIPT_FILENAME' ]);
637        $ path = $ base . '/' .$ path ;
638        if ($ run == 0 ) { //  избегаем бесконечной рекурсии, когда база по
```

```
какой-то причине не является абсолютной
639       $ запустить ++;
640       вернуть полный путь ($ path , $ existences );
641          }
642     }
643   $ запуск = 0 ;
644
645   // канонизировать
646     $ path = взорвать ( '/' , $ path );
647   $ новый_путь = [];
648     foreach ($ path  как $ p ) {
649     если ($ p === '' || $ p === '.' ) продолжить ;
650     если ($ p === '..' ) {
651              array_pop ($ newpath );
652       продолжать ;
653          }
654          $ newpath [] = $ p ;
655     }
656   $ окончательный путь = $ корень . implode ( '/' , $ newpath );
657
658   // проверка существования при необходимости (кроме случаев модульного
тестирования)
659   если ($ существует && ! определено ( 'DOKU_UNITTEST' ) && ! file_exists ($
finalpath )) {
660     вернуть ложь ;
661     }
662   вернуть $ finalpath ;
663}
664
```