| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [TFLayers/src](http://docs.google.com/index.html) - TF.cpp (source / [functions](http://docs.google.com/TF.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 42 | 64 | 65.6 % | | Date: | 2012-03-23 |  | Functions: | 16 | 20 | 80.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 0 | 6 | 0.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : :   2 : : #include "ConverterParams.h"  3 : : #include "ErrorHandler.h"  4 : : #include "SimplePattern.h"  5 : : #include "TF.h"  6 : : #include "SDDB.h"  7 : : #include "HealthCheckCounter.h"  8 : :   9 : : namespace nft {  10 : :   11 : : TF::TF()  12 : 10423 : : mShotList( ConverterParams::instance()  13 : : ->getStartupMessage()->usConcurrency ),  14 : : mParentSF(),  15 : : mLocX(),  16 : : mLocY(),  17 : : mRow(),  18 : : mColumn(),  19 : : mPatCode(),  20 : : mDistributed(),  21 : 10423 : mDiscarded() {  22 : : }  23 : :   24 : :   25 : 10423 : TF::~TF() {  26 : : }  27 : :   28 : : TF::TF(const TF & source)  29 : 0 : : mShotList( source.mShotList ),  30 : : mParentSF(source.mParentSF),  31 : : mLocX(source.mLocX),  32 : : mLocY(source.mLocY),  33 : : mRow(source.mRow),  34 : : mColumn(source.mColumn),  35 : : mPatCode(source.mPatCode),  36 : : mDistributed(source.mDistributed),  37 : 0 : mDiscarded(source.mDiscarded) {  38 : : }  39 : :   40 : : TF &  41 : 0 : TF::operator=(const TF & source) {  42 [ # # ]: 0 : if( this != &source )  43 : : {  44 : 0 : mShotList = source.mShotList ;  45 : 0 : mParentSF = source.mParentSF ;  46 : 0 : mLocX = source.mLocX ;  47 : 0 : mLocY = source.mLocY ;  48 : 0 : mRow = source.mRow ;  49 : 0 : mColumn = source.mColumn ;  50 : 0 : mPatCode = source.mPatCode ;  51 : 0 : mDistributed = source.mDistributed ;  52 : 0 : mDiscarded = source.mDiscarded;  53 : : }  54 : 0 : return \*this;  55 : : }  56 : :   57 : : void  58 : 0 : TF::allocateShotList() {  59 : : }  60 : :   61 : : void  62 : : TF::addShot(const SimplePattern & inShot,  63 : 15956 : ThreadID inThreadID) throw(DPL::Error \*) {  64 : :   65 : 15956 : HealthCheckCounter \*pHealthChechCnt = HealthCheckCounter::instance();  66 : 15956 : pHealthChechCnt->incrCounter(HealthCheckCounter::eUpdateFrequent);  67 : 15956 : SDDB \*pSDDB = SDDB::instance();  68 : 15956 : try  69 : : {  70 : 15956 : pSDDB->shotAdded();  71 : : }  72 [ # # ]: 0 : catch(DPL::Error \*pChild)  73 : : {  74 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  75 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_,  76 : : \_\_LINE\_\_, ErrorHandler::eErrFunction,  77 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  78 : : "SDDB::shotAdded"));  79 : : }  80 : 15956 : SimplePattern \*pMem = 0;  81 : 15956 : try  82 : : {  83 : 15956 : pMem = mShotList.getMemoryForShots( inThreadID );  84 : 15956 : new(pMem) SimplePattern(inShot);  85 : : }  86 [ # # ]: 0 : catch( DPL::Error \*pChild )  87 : : {  88 : 0 : Logger::instance()->log(LogMessageTable::ePATTERN\_ALLOC,  89 : : "Memory allocation failed for Shot");  90 : :   91 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  92 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  93 : : ErrorHandler::eErrFunction,  94 : : pErr->getErrorMsg(  95 : : ErrorHandler::eErrFunction,  96 : : "SimplePatternBuffer::getMemoryForShots"));  97 : : }  98 : : }  99 : :   100 : : void  101 : 10535 : TF::releaseShotMem() {  102 : 10535 : mShotList.releaseMemory();  103 : : }  104 : :   105 : : void  106 : : TF::resetTF()  107 : 10151 : {  108 : 10151 : mParentSF = NULL;  109 : 10151 : mDistributed = false;  110 : 10151 : mLocX = 0;  111 : 10151 : mLocY = 0;  112 : 10151 : mDiscarded = false;  113 : 10151 : mRow = 0;  114 : 10151 : mColumn = 0;  115 : 10151 : mPatCode = 0;  116 : : }  117 : :   118 : : void  119 : 10472 : TF::setParentSF( SubField \*inValue ) {  120 : 10472 : mParentSF = inValue;  121 : : }  122 : :   123 : : void  124 : 10295 : TF::setDistributed( bool inDistFlag ) {  125 : 10295 : mDistributed = inDistFlag;  126 : : }  127 : :   128 : :   129 : : void  130 : 10472 : TF::setLocX(int32\_t inLocx) {  131 : 10472 : mLocX = inLocx;  132 : : }  133 : :   134 : : void  135 : 10472 : TF::setLocY(int32\_t inLocy) {  136 : 10472 : mLocY = inLocy;  137 : : }  138 : :   139 : : void  140 : 10295 : TF::setDiscarded(bool inValue) {  141 : 10295 : mDiscarded = inValue;  142 : : }  143 : :   144 : : void  145 : 10436 : TF::setRow(uint32\_t inValue) {  146 : 10436 : mRow = inValue;  147 : : }  148 : :   149 : : void  150 : 10436 : TF::setColumn(uint32\_t inValue) {  151 : 10436 : mColumn = inValue;  152 : : }  153 : :   154 : : void  155 : 10535 : TF::setPatCode(PatternCode inValue) {  156 : 10535 : mPatCode = inValue;  157 : : }  158 : :   159 : 1215 : } // namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |