Comparison of the ndarray repository

Summary of Repositories

Comparison run at 07:32PM on June 07, 2015
There are **17** differences between the two repositories

Repository **/Users/nate/repos_hsc/ndarray/**
Revision **1e102dd4f775e3c6afcdc5930096552123c3689d**
Branch **master**
Last commit was on **2013-12-06 16:06:11 -0500**

Repository **/Users/nate/repos_lsst/ndarray/**
Revision **7408a83a3aa3df1509bb22d61aa4438c65cef2b6**
Branch **master**
Last commit was on **2015-01-22 12:09:07 -0500**

Files only in /Users/nate/repos_hsc/ndarray/

Files only in /Users/nate/repos_lsst/ndarray/

List of the files in common

Files without links do not differ

- `include/ndarray/detail/Core.h`
- `doc/doxygen.conf.in`

- include/ndarray/swig/eigen.h
- ups/ndarray.build
- doc/SConscript
- include/ndarray/fft/FFTWTraits.h.m4
- include/ndarray/ExpressionTraits.h
- include/ndarray/ArrayBaseN.h.m4
- SConstruct
- include/ndarray/detail/BinaryOp.h
- include/ndarray/casts.h
- include/ndarray/fft/FourierOps.h
- include/ndarray/Array.h
- include/ndarray.h
- tests/swig_test_mod.i
- include/ndarray/arange.h
- include/ndarray/detail/StridedIterator.h
- include/ndarray/formatting.h
- include/ndarray/vectorize.h
- .gitignore
- include/ndarray/swig/ufunctors.h
- include/ndarray/tables_fwd.h
- include/ndarray/swig.h
- include/ndarray/fft/FourierTransform.h
- include/ndarray/ArrayRef.h.m4
- include/ndarray/views.h
- include/ndarray/fft/FourierTransform.cc
- ups/ndarray.cfg
- include/ndarray/eigen_fwd.h
- include/ndarray/detail/ArrayAccess.h
- include/ndarray/swig/numpy.h
- include/ndarray/operators.h
- tests/SConscript
- include/ndarray_fwd.h
- include/ndarray/fft.h
- include/ndarray/detail/ViewBuilder.h
- include/ndarray/fft_fwd.h
- tests/ndarray-python-mod.cc
- include/ndarray/operators.h.m4
- include/ndarray/initialization.h
- tests/ndarray-python.py

- `include/SConscript`
- `include/ndarray/fft/FourierTraits.h`
- `TODO`
- `tests/ndarray-eigen.cc`
- `include/ndarray/swig/Vector.h`
- `include/ndarray/swig/PyConverter.h`
- `python/ndarray.i`
- `include/ndarray/ArrayTraits.h`
- `include/ndarray/ArrayBase.h`
- `ups/ndarray.table`
- `include/ndarray/detail/UnaryOp.h`
- `include/ndarray/ExpressionBase.h`
- `include/ndarray/types.h`
- `include/ndarray/eigen.h`
- `include/ndarray/Manager.h`
- `tests/ndarray-fft.cc`
- `include/ndarray/detail/NestedIterator.h`
- `tests/ndarray.cc`
- `include/ndarray/Vector.h.m4`

include/ndarray/Array.h

Diff:

```
                // -*- c++ -*-
                /*
                 * LSST Data Management System
                 * Copyright 2008, 2009, 2010, 2011 LSST Corporation.
                 *
                 * This product includes software developed by the
                 * LSST Project (http://www.lsst.org/).
                 *
                 * This program is free software: you can redistribute it and/or modi
fy
                 * it under the terms of the GNU General Public License as published
by
                 * the Free Software Foundation, either version 3 of the License, or
                 * (at your option) any later version.
                 *
                 * This program is distributed in the hope that it will be useful,
```

```
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the LSST License Statement and
 * the GNU General Public License along with this program.  If not,
 * see .
 */
#ifndef NDARRAY_Array_h_INCLUDED
#define NDARRAY_Array_h_INCLUDED

/**
 *  @file ndarray/Array.h
 *
 *  @brief Definitions for Array.
 */

#include "ndarray_fwd.h"
#include "ndarray/ArrayTraits.h"
#include "ndarray/ArrayBaseN.h"
#include "ndarray/Vector.h"
#include "ndarray/detail/Core.h"
#include "ndarray/views.h"

namespace ndarray {

/**
 *  @brief A multidimensional strided array.
 *
 *  Array is the workhorse class of the ndarray library.
 */
template
class Array : public ArrayBaseN< Array > {
    typedef ArrayBaseN Super;
    typedef typename Super::Core Core;
    typedef typename Super::CorePtr CorePtr;
public:

    /**
     *  @brief Default constructor.
     *
     *  Creates an empty array with zero dimensions and null memory.
     */
    Array() : Super(0, Core::create()) {}
```

```
    /**
     *  @brief Non-converting copy constructor.
     */
    Array(Array const & other) : Super(other._data, other._core) {}

    /**
     *  @brief Converting copy constructor.
     *
     *  Implicit conversion is allowed for non-const to const and for

     *  more guaranteed RMC to less guaranteed RMC (see \ref index).
     */
    template
    Array(
        Array const & other
#ifndef DOXYGEN
        , typename boost::enable_if,void*>::type=0
#endif
    ) : Super(other._data, other._core) {}

    /**
     *  @brief Converting copy constructor.
     *
     *  Implicit conversion is allowed for non-const to const and for

     *  more guaranteed RMC to less guaranteed RMC (see \ref index).
     */
    template
    Array(
        ArrayRef const & other
#ifndef DOXYGEN
        , typename boost::enable_if,void*>::type=0
#endif
    ) : Super(other._data, other._core) {}

    /**
     *  @brief Non-converting shallow assignment.
     */
    Array & operator=(Array const & other) {
        if (&other != this) {
            this->_data = other._data;
            this->_core = other._core;
        }
        return *this;
    }
```

```
                    /**
                     *  @brief Converting shallow assignment.
                     *
                     *  Implicit conversion is allowed for non-const -> const and for

                     *  more guaranteed RMC -> less guaranteed RMC (see \ref index).
                     */
                    template
                #ifndef DOXYGEN
                    typename boost::enable_if, Array &>::type
                #else
                    Array &
                #endif
                    operator=(Array const & other) {
                        this->_data = other._data;
                        this->_core = other._core;
                        return *this;
                    }

                    /**
                     *  @brief Converting shallow assignment.
                     *
                     *  Implicit conversion is allowed for non-const -> const and for

                     *  more guaranteed RMC -> less guaranteed RMC (see \ref index).
                     */
                    template
                #ifndef DOXYGEN
                    typename boost::enable_if, Array &>::type
                #else
                    Array &
                #endif
                    operator=(ArrayRef const & other) {
                        this->_data = other._data;
                        this->_core = other._core;
                        return *this;
                    }

                    /**
                     *  @brief Shallow equality comparison: return true if the arrays
share data and
                     *         have the same shape and strides.
                     */
                    template
```

```
                    bool operator==(Array const & other) const {
                        return this->getData() == other.getData()
                            && this->getShape() == other.getShape()
                            && this->getStrides() == other.getStrides();
                    }

                    /**
                     *  @brief Shallow inequality comparison.
                     */
                    template
                    bool operator!=(Array const & other) const {
                        return !this->operator==(other);
                    }

                    /// @brief Lightweight shallow swap.
                    void swap(Array & other) {
                        std::swap(this->_data, other._data);
                        this->_core.swap(other._core);
                    }

                    /**
                     *  @brief Return true if the Array is definitely unique.
                     *
                     *  This will only return true if the manager overrides Manager::
isUnique();
                     *  this is true for the SimpleManager used by ndarray::allocate,
 but it is
                     *  not true for ExternalManager.
                     */
                    bool isUnique() const { return this->_core->isUnique(); }

                private:
                    template  friend class Array;
                    template  friend class ArrayRef;
                    template  friend struct ArrayTraits;
                    template  friend class ArrayBase;
```

```
179  ^9ccb4c9 -     template  friend struct detail::ArrayAccess;
             ?                                        ^^^^^
179  33800a4f +     template  friend class detail::ArrayAccess;
```

```
            ?                                          +++ ^

                /// @internal @brief Construct an Array from a pointer and Core.
                Array(T * data, CorePtr const & core) : Super(data, core) {}
            };

            } // namespace ndarray

            #endif // !NDARRAY_Array_h_INCLUDED
```

Return to list

Commits in /Users/nate/repos_hsc/ndarray/

^9ccb4c9

```
```

Commits in /Users/nate/repos_lsst/ndarray/

33800a4f

```
commit 33800a4f53b5a67c85c566a894d81511faf2681d
Author: Jim Bosch
Date:   Fri Jan 24 15:15:08 2014 -0500

    Fix clang warnings about class/struct mismatches (#3081)
```

Return to list

include/ndarray/initialization.h

Diff:

```
                // -*- c++ -*-
                /*
                 * LSST Data Management System
                 * Copyright 2008, 2009, 2010, 2011 LSST Corporation.
                 *
                 * This product includes software developed by the
                 * LSST Project (http://www.lsst.org/).
                 *
                 * This program is free software: you can redistribute it and/or modi
fy
                 * it under the terms of the GNU General Public License as published
by
                 * the Free Software Foundation, either version 3 of the License, or
                 * (at your option) any later version.
                 *
                 * This program is distributed in the hope that it will be useful,
                 * but WITHOUT ANY WARRANTY; without even the implied warranty of
                 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
                 * GNU General Public License for more details.
                 *
                 * You should have received a copy of the LSST License Statement and
                 * the GNU General Public License along with this program.  If not,
                 * see .
                 */
                #ifndef NDARRAY_initialization_h_INCLUDED
                #define NDARRAY_initialization_h_INCLUDED

                /**
                 *  \file ndarray/initialization.h @brief Construction functions for
array.
                 */

                #include "ndarray/Array.h"
                #include "ndarray/Manager.h"

                namespace ndarray {
                namespace detail {

                struct NullOwner {};

                template
                class Initializer {
                public:

                    template
```

```
                    operator Array () const {
                        return static_cast(this)->template apply< Array >();
                    }

                    template
                    operator ArrayRef () const {
                        return static_cast(this)->template apply< ArrayRef >();
                    }

                };

                template
                class SimpleInitializer : public Initializer< N, SimpleInitializer >
{
                public:

                    template
                    Target apply() const {
                        typedef detail::ArrayAccess< Target > Access;
                        typedef typename Access::Core Core;
                        typedef typename Access::Element Element;
                        DataOrderEnum order = (ExpressionTraits< Target >::RMC::value
 < 0) ? COLUMN_MAJOR : ROW_MAJOR;
                        int total = _shape.product();
                        std::pair p = SimpleManager::allocate(total);
                        return Access::construct(p.second, Core::create(_shape, order
, p.first));
                    }

                    explicit SimpleInitializer(Vector const & shape) : _shape(shape)
{}

                private:
                    Vector _shape;
                };

                template
                class ExternalInitializer : public Initializer< N, ExternalInitialize
r > {
                public:

                    template
                    Target apply() const {
                        typedef detail::ArrayAccess< Target > Access;
                        typedef typename Access::Core Core;
```

```
83   d7e2d1b0 -         typedef typename Access::Element Element;

                        Manager::Ptr manager;
                        if (!boost::is_same::value) {
                            manager = makeManager(_owner);
                        }
                        return Access::construct(_data, Core::create(_shape, _strides
, manager));
                    }

                    ExternalInitializer(
                        T * data,
                        Vector const & shape,
                        Vector const & strides,
                        Owner const & owner
                    ) : _data(data), _owner(owner), _shape(shape), _strides(strides)
{}

                private:
                    T * _data;
                    Owner _owner;
                    Vector _shape;
                    Vector _strides;
                };

                } // namespace detail

                /// @addtogroup MainGroup
                /// @{

                /**
                 *  @brief Create an expression that allocates uninitialized memory f
or an array.
                 *
                 *  @returns A temporary object convertible to an Array with fully co
ntiguous row-major strides.
                 */
                template
                inline detail::SimpleInitializer allocate(Vector const & shape) {
                    return detail::SimpleInitializer(shape);
                }

                /**
```

```
                 *  @brief Create an expression that allocates uninitialized memory f
or a 1-d array.
                 *
                 *  @returns A temporary object convertible to an Array with fully co
ntiguous row-major strides.
                 */
                inline detail::SimpleInitializer<1> allocate(int n) {
                    return detail::SimpleInitializer<1>(ndarray::makeVector(n));
                }

                /**
                 *  @brief Create an expression that allocates uninitialized memory f
or a 2-d array.
                 *
                 *  @returns A temporary object convertible to an Array with fully co
ntiguous row-major strides.
                 */
                inline detail::SimpleInitializer<2> allocate(int n1, int n2) {
                    return detail::SimpleInitializer<2>(ndarray::makeVector(n1, n2));

                }

                /**
                 *  @brief Create an expression that allocates uninitialized memory f
or a 3-d array.
                 *
                 *  @returns A temporary object convertible to an Array with fully co
ntiguous row-major strides.
                 */
                inline detail::SimpleInitializer<3> allocate(int n1, int n2, int n3)
{
                    return detail::SimpleInitializer<3>(ndarray::makeVector(n1, n2, n
3));
                }

                /**
                 *  @brief Create a new Array by copying an Expression.
                 */
                template
                inline ArrayRef::type,
                                Derived::ND::value, Derived::ND::value>
                copy(ExpressionBase const & expr) {
                    ArrayRef::type,
                        Derived::ND::value,Derived::ND::value> r(
                            allocate(expr.getShape())
```

```
                        );
                    r = expr;
                    return r;
                }

                /// @brief Compute row- or column-major strides for the given shape.
                template
                Vector computeStrides(Vector const & shape, DataOrderEnum order=ROW_M
AJOR) {
                    Vector r(1);
                    if (order == ROW_MAJOR) {
                        for (int n=N-1; n > 0; --n) r[n-1] = r[n] * shape[n];
                    } else {
                        for (int n=1; n < N; ++n) r[n] = r[n-1] * shape[n-1];
                    }
                    return r;
                }

                /**
                 *  @brief Create an expression that initializes an Array with extern
ally allocated memory.
                 *
                 *  No checking is done to ensure the shape, strides, and data pointe
rs are sensible.
                 *
                 *  @param[in] data     A raw pointer to the first element of the Arr
ay.
                 *  @param[in] shape    A Vector of dimensions for the new Array.
                 *  @param[in] strides  A Vector of strides for the new Array.
                 *  @param[in] owner    A copy-constructable object with an internal
reference count
                 *                      that owns the memory pointed at by 'data'.
                 *
                 *  @returns A temporary object convertible to an Array.
                 */
                template
                inline detail::ExternalInitializer external(
                    T * data,
                    Vector const & shape,
                    Vector const & strides,
                    Owner const & owner
                ) {
                    return detail::ExternalInitializer(data, shape, strides, owner);
                }
```

```
                /**
                 *  @brief Create an expression that initializes an Array with extern
ally allocated memory.
                 *
                 *  No checking is done to ensure the shape, strides, and data pointe
rs are sensible.  Memory will not
                 *  be managed at all; the user must ensure the data pointer remains
valid for the lifetime of the array.
                 *
                 *  @param[in] data     A raw pointer to the first element of the Arr
ay.
                 *  @param[in] shape    A Vector of dimensions for the new Array.
                 *  @param[in] strides  A Vector of strides for the new Array.
                 *
                 *  @returns A temporary object convertible to an Array.
                 */
                template
                inline detail::ExternalInitializer external(
                    T * data,
                    Vector const & shape,
                    Vector const & strides
                ) {
                    return detail::ExternalInitializer(data, shape, strides, detail::
NullOwner());
                }

                /**
                 *  @brief Create an expression that initializes an Array with extern
ally allocated memory.
                 *
                 *  No checking is done to ensure the shape and data pointers are sen
sible.
                 *
                 *  @param[in] data     A raw pointer to the first element of the Arr
ay.
                 *  @param[in] shape    A Vector of dimensions for the new Array.
                 *  @param[in] order    Whether the strides are row- or column-major.
                 *  @param[in] owner    A copy-constructable object with an internal
reference count
                 *                      that owns the memory pointed at by 'data'.
                 *
                 *  @returns A temporary object convertible to an Array.
                 */
                template
                inline detail::ExternalInitializer external(
```

```
                    T * data,
                    Vector const & shape,
                    DataOrderEnum order,
                    Owner const & owner
                ) {
                    return detail::ExternalInitializer(data, shape, computeStrides(sh
ape, order), owner);
                }

                /**
                 *  @brief Create an expression that initializes an Array with extern
ally allocated memory.
                 *
                 *  No checking is done to ensure the shape and data pointers are sen
sible.  Memory will not
                 *  be managed at all; the user must ensure the data pointer remains
valid for the lifetime of the array.
                 *
                 *  @param[in] data     A raw pointer to the first element of the Arr
ay.
                 *  @param[in] shape    A Vector of dimensions for the new Array.
                 *  @param[in] order    Whether the strides are row- or column-major.
                 *
                 *  @returns A temporary object convertible to an Array.
                 */
                template
                inline detail::ExternalInitializer external(
                    T * data,
                    Vector const & shape,
                    DataOrderEnum order = ROW_MAJOR
                ) {
                    return detail::ExternalInitializer(
                        data, shape, computeStrides(shape, order), detail::NullOwner(
)
                    );
                }

                /// @}

                } // namespace ndarray

                #endif // !NDARRAY_initialization_h_INCLUDED
```

Return to list

Commits in /Users/nate/repos_hsc/ndarray/

d7e2d1b0

```
commit d7e2d1b0e1cc215ae737eb47cd13bb865aafce5e
Author: jbosch
Date:   Wed Feb 16 09:00:54 2011 +0000

    ndarray - synced from extrnal (added tables, switch to custom Manager clas
s instead of shared_ptr ownership)
```

Commits in /Users/nate/repos_lsst/ndarray/

Return to list

include/ndarray/ArrayBase.h

Diff:

```
                // -*- c++ -*-
                /*
                 * LSST Data Management System
                 * Copyright 2008, 2009, 2010, 2011 LSST Corporation.
                 *
                 * This product includes software developed by the
                 * LSST Project (http://www.lsst.org/).
                 *
                 * This program is free software: you can redistribute it and/or modi
fy
                 * it under the terms of the GNU General Public License as published
by
                 * the Free Software Foundation, either version 3 of the License, or
                 * (at your option) any later version.
                 *
                 * This program is distributed in the hope that it will be useful,
                 * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the LSST License Statement and
 * the GNU General Public License along with this program.  If not,
 * see .
 */
#ifndef NDARRAY_ArrayBase_h_INCLUDED
#define NDARRAY_ArrayBase_h_INCLUDED

/**
 *  @file ndarray/ArrayBase.h
 *
 *  @brief Definitions for ArrayBase.
 */


#include

#include "ndarray/ExpressionBase.h"
#include "ndarray/Vector.h"
#include "ndarray/detail/Core.h"
#include "ndarray/detail/NestedIterator.h"
#include "ndarray/detail/StridedIterator.h"
#include "ndarray/detail/ArrayAccess.h"
#include "ndarray/detail/ViewBuilder.h"
#include "ndarray/ArrayTraits.h"
#include "ndarray/eigen_fwd.h"

namespace ndarray {

/**
 *  @class ArrayBase
 *  @brief CRTP implementation for Array and ArrayRef.
 *
 *  @ingroup MainGroup
 *
 *  Implements member functions that need specialization for 1D array
s.
 */
template
class ArrayBase : public ExpressionBase {
protected:
    typedef ExpressionTraits Traits;
    typedef typename Traits::Core Core;
```

```
                    typedef typename Traits::CorePtr CorePtr;
                public:
                    /// @brief Data type of array elements.
                    typedef typename Traits::Element Element;
                    /// @brief Nested array or element iterator.
                    typedef typename Traits::Iterator Iterator;
                    /// @brief Nested array or element reference.
                    typedef typename Traits::Reference Reference;
                    /// @brief Nested array or element value type.
                    typedef typename Traits::Value Value;
                    /// @brief Number of dimensions (boost::mpl::int_).
                    typedef typename Traits::ND ND;
                    /// @brief Number of guaranteed row-major contiguous dimensions,
counted from the end (boost::mpl::int_).
                    typedef typename Traits::RMC RMC;
                    /// @brief Vector type for N-dimensional indices.
                    typedef Vector Index;
                    /// @brief ArrayRef to a reverse-ordered contiguous array; the re
sult of a call to transpose().
                    typedef ArrayRef FullTranspose;
                    /// @brief ArrayRef to a noncontiguous array; the result of a cal
l to transpose(...).
                    typedef ArrayRef Transpose;
                    /// @brief The corresponding Array type.
                    typedef Array Shallow;
                    /// @brief The corresponding ArrayRef type.
                    typedef ArrayRef Deep;

                    /// @brief Return a single subarray.
                    Reference operator[](int n) const {
                        return Traits::makeReference(
                            this->_data + n * this->template getStride<0>(),
                            this->_core
                        );
                    }

                    /// @brief Return a single element from the array.
                    Element & operator[](Index const & i) const {
                        return *(this->_data + this->_core->template computeOffset(i)
);
                    }

                    /// @brief Return an Iterator to the beginning of the array.
                    Iterator begin() const {
                        return Traits::makeIterator(
```

```
                            this->_data,
                            this->_core,
                            this->template getStride<0>()
                        );
                    }

                    /// @brief Return an Iterator to one past the end of the array.
                    Iterator end() const {
                        return Traits::makeIterator(
                            this->_data + this->template getSize<0>() * this->templat
e getStride<0>(),
                            this->_core,
                            this->template getStride<0>()
                        );
                    }

                    /// @brief Return a raw pointer to the first element of the array
.
                    Element * getData() const { return _data; }

                    /// @brief Return true if the array has a null data point.
                    bool isEmpty() const { return _data == 0; }

                    /// @brief Return the opaque object responsible for memory manage
ment.
                    Manager::Ptr getManager() const { return this->_core->getManager(
); }

                    /// @brief Return the size of a specific dimension.
                    template  int getSize() const {
                        return detail::getDimension

(*this->_core).getSize();
                    }

                    /// @brief Return the stride in a specific dimension.
                    template  int getStride() const {
                        return detail::getDimension

(*this->_core).getStride();
                    }

                    /// @brief Return a Vector of the sizes of all dimensions.
                    Index getShape() const { Index r; this->_core->fillShape(r); retu
rn r; }
```

```
                    /// @brief Return a Vector of the strides of all dimensions.
                    Index getStrides() const { Index r; this->_core->fillStrides(r);
return r; }

                    /// @brief Return the total number of elements in the array.
                    int getNumElements() const { return this->_core->getNumElements()
; }

                    /// @brief Return a view of the array with the order of the dimen
sions reversed.
                    FullTranspose transpose() const {
                        Index shape = getShape();
                        Index strides = getStrides();
                        for (int n=0; n < ND::value / 2; ++n) {
                            std::swap(shape[n], shape[ND::value-n-1]);
                            std::swap(strides[n], strides[ND::value-n-1]);
                        }
                        return FullTranspose(
                            getData(),
                            Core::create(shape, strides, getManager())
                        );
                    }

                    /// @brief Return a view of the array with the dimensions permute
d.
                    Transpose transpose(Index const & order) const {
                        Index newShape;
                        Index newStrides;
                        Index oldShape = getShape();
                        Index oldStrides = getStrides();
                        for (int n=0; n < ND::value; ++n) {
                            newShape[n] = oldShape[order[n]];
                            newStrides[n] = oldStrides[order[n]];
                        }
                        return Transpose(
                            getData(),
                            Core::create(newShape, newStrides, getManager())
                        );
                    }

                    /// @brief Return a Array view to this.
                    Shallow const shallow() const { return Shallow(this->getSelf());
}
```

```
                    /// @brief Return an ArrayRef view to this.
                    Deep const deep() const { return Deep(this->getSelf()); }

                    //@{
                    /**
                     *  @name Eigen3 Interface
                     *
                     *  These methods return Eigen3 views to the array.  Template
                     *  parameters optionally control the expression type (Matrix/Arr
ay) and
                     *  the compile-time dimensions.
                     *
                     *  The inline implementation is included by ndarray/eigen.h.
                     */
                    template
                    EigenView asEigen() const;

                    template
                    EigenView asEigen() const;

                    template
                    EigenView asEigen() const;

                    EigenView asEigen() const;
                    //@}

                    /// @brief A template metafunction class to determine the result
of a view indexing operation.
                    template
                    struct ResultOf {
                        typedef Element Element_;
                        typedef typename detail::ViewTraits::ND ND_;
                        typedef typename detail::ViewTraits::RMC RMC_;
                        typedef ArrayRef Type;
                        typedef Array Value;
                    };

                    /// @brief Return a general view into this array (see @ref ndarra
yTutorial).
                    template
                    typename ResultOf< View >::Type
                    operator[](View const & def) const {
                        return detail::buildView(this->getSelf(), def._seq);
                    }
```

```
                protected:
                    template  friend class Array;
                    template  friend class ArrayRef;
                    template  friend struct ArrayTraits;
                    template  friend class detail::NestedIterator;
                    template  friend class ArrayBase;

225  ^9ccb4c9 -     template  friend struct detail::ArrayAccess;
             ?                                          ^^^^^
225  33800a4f +     template  friend class detail::ArrayAccess;
             ?                                         +++ ^

                    Element * _data;
                    CorePtr _core;

                    void operator=(ArrayBase const & other) {
                        _data = other._data;
                        _core = other._core;
                    }

                    ArrayBase(Element * data, CorePtr const & core) : _data(data), _c
ore(core) {}
                };

                } // namespace ndarray

                #endif // !NDARRAY_ArrayBase_h_INCLUDED
```

Return to list

Commits in /Users/nate/repos_hsc/ndarray/

^9ccb4c9

Commits in /Users/nate/repos_lsst/ndarray/

33800a4f

```
commit 33800a4f53b5a67c85c566a894d81511faf2681d
Author: Jim Bosch
Date:   Fri Jan 24 15:15:08 2014 -0500

    Fix clang warnings about class/struct mismatches (#3081)
```

Return to list

ups/ndarray.table

Diff:

```
1    7f3065b0 - setupRequired(boost >= 1.47.0)
           ?                     ----------
1    6ad128e5 + setupRequired(boost)
             setupRequired(python)
3    fb4d216b - setupRequired(numpy >= 1.5)
           ?                     -------
3    6ad128e5 + setupRequired(numpy)
4    df988f4b - setupRequired(eigen >= 3.0.0)
```

```
            ?                       ---------

4    6ad128e5 + setupRequired(eigen)

5    7f3065b0 - setupRequired(base >= 4.6.0.0)

            ?                       -----------

5    6ad128e5 + setupRequired(base)

              setupRequired(swig)

7    fb4d216b - setupRequired(fftw >= 3.1)

            ?                       -------

7    6ad128e5 + setupRequired(fftw)


9    955f9cb7 - envAppend(PYTHONPATH, ${PRODUCT_DIR}/python)

            ?     ^^

9    82b9a38f + envPrepend(PYTHONPATH, ${PRODUCT_DIR}/python)

            ?     ^^^
```

Return to list

Commits in /Users/nate/repos_hsc/ndarray/

7f3065b0

```
commit 7f3065b0b1db306ee96118e760d122fb6cc2ac2e
Author: jbosch
Date:   Tue Oct 18 21:44:53 2011 +0000

    #1780 - lots of dependency tree fixes; removed separate scons package
```

fb4d216b

```
commit fb4d216bb771637fbcfab73988414b6890c98785
Author: rhl
Date:   Mon Nov 7 21:44:13 2011 +0000

    Relax python/numpy/fftw requirements
```

df988f4b

```
commit df988f4bf8d582d775203725ce098234851469e5
Author: jbosch
Date:   Mon Sep 26 18:58:31 2011 +0000

    ndarray #1752 - synced with upstream ndarray eigen3 branch
```

955f9cb7

```
commit 955f9cb7356d9fa1151fcfa93ca8aeccb64abedd
Author: Jim Bosch
Date:   Wed Nov 16 22:47:31 2011 -0500

    added support for git-based version introspection
```

Commits in /Users/nate/repos_lsst/ndarray/

6ad128e5

```
commit 6ad128e5a208d54410538ba298c44fc70fed79c0
Author: Mario Juric
Date:   Wed Mar 5 16:29:24 2014 -0600

    removed explicit versions from the table file.
```

82b9a38f

```
commit 82b9a38f2aabc061dfe06d0971f1c3726f5f7d4c
Author: Robyn Allsman
Date:   Fri Nov 14 20:52:45 2014 -0600

    Replace envAppend with envPrepend in ups table files.
```

Return to list