NAME

vec_t, cvec_t – Data Vector Classes

SYNOPSIS

```
#include <vec_t.h>

class cvec_t {
    friend class vec_t; // so vec_t can look at VEC_t
public:
    cvec_t();
    cvec_t(const cvec_t& v1, const cvec_t& v2);
    cvec_t(const void* p, size_t l);
    cvec_t(const cvec_t& v, size_t offset, size_t limit);

    ~cvec_t();

    void split(size_t l1, cvec_t& v1, cvec_t& v2);
    cvec_t& put(const cvec_t& v, size_t offset, size_t nbytes);
    cvec_t& put(const void* p, size_t l);
    cvec_t& put(const cvec_t& v);
    cvec_t& reset();
    cvec_t& set(const cvec_t& v1, const cvec_t& v2);
    cvec_t& set(const cvec_t& v);

    cvec_t& set(const void* p, size_t l);
    cvec_t& set(const cvec_t& v, size_t offset, size_t limit);

    size() const;
    size_t copy_to(void* p, size_t limit = 0x7fffffff) const;
    int cmp(const cvec_t& v, size_t* common_size = 0) const;
    int cmp(const void* s, size_t len) const;

    static int cmp(const cvec_t& v1, const cvec_t& v2, size_t* common_size = 0)
    int count() const;

    int checksum() const;
    void calc_kvl(uint4& h) const;
    void init();

    is_pos_inf() const;
    is_neg_inf() const;

    friend inline bool operator<(const cvec_t& v1, const cvec_t& v2);
    friend inline bool operator<=(const cvec_t& v1, const cvec_t& v2);
    friend inline bool operator>=(const cvec_t& v1, const cvec_t& v2);
    friend inline bool operator>(const cvec_t& v1, const cvec_t& v2);
    friend inline bool operator==(const cvec_t& v1, const cvec_t& v2);
    friend inline bool operator!=(const cvec_t& v1, const cvec_t& v2);

    static cvec_t pos_inf;
    static cvec_t neg_inf;

class vec_t : public cvec_t {
public:
```

```
        vec_t();
        vec_t(const cvec_t& v1, const cvec_t& v2);
        vec_t(const void* p, size_t l);
        vec_t(const vec_t& v, size_t offset, size_t limit);

        /*
         *  copy_from() does not change vec_t itself, but overwrites
         *  the data area to which the vec points
         *  (temporarily made const for VAS compatibility)
         */
        const vec_t& copy_from(
          const void* p,
          size_t limit,
          size_t offset = 0) const;   // offset tells where
                                 //in the vec to begin to copy

        vec_t& copy_from(const cvec_t& v);
        vec_t& copy_from(
          const cvec_t& v,
          size_t offset,          // offset in v
          size_t limit,           // # bytes
          size_t myoffset = 0);   // offset in this

        void*   ptr(int index) const;
        size_t  len(int index) const;

        static vec_t& pos_inf;
        static vec_t& neg_inf;
    };
```

DESCRIPTION

TODO

VERSION

This manual page applies to Version 2.0 of the Shore Storage Manager.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

SEE ALSO