CDF Issues

Will we still have a lab this week?

Yes, It is our last one. It is posted on cslab

www.cs.toronto.edu/~mcraig/209/index.html

Will you extend due dates?

Yes. The lab is due Monday (instead of Friday)

A4 is due Wednesday (the last possible day) instead of Monday.

A Terrific TA or Grad student instructor?

If you have an excellent TA or grad student instructor this term, please consider nominating them for an award.

uoft.me/ta-awards

Bit arrays

King: 20.1, 20.2

`select` example from week 10

```
fd_set rfds;
struct timeval tv;
int retval;

FD_ZERO(&rfds);  /* Watch stdin (fd 0) for input */
FD_SET(STDIN_FILENO, &rfds);
tv.tv_sec = 5;   /* Wait up to five seconds. */
tv.tv_usec = 0;
retval = select(1, &rfds, NULL, NULL, &tv);
if (retval == -1)
  perror("select()");
else if (retval > 0)
  printf("Data is available now.\n");
  /* FD_ISSET(0, &rfds) will be true, can use read() */
else
  printf("No data within five seconds.\n");
```

Bit strings

- Signal mask and file descriptor sets are implemented using bit array or bit strings.
- You should always use the supplied functions macros to manipulate these structures.
- It is useful to know how they are implemented.
- Each bit represents an element of the set

1	in the set
0	not in the set

Bitwise operators

- shift (note that bits fall off the ends)

  << left shift

  >> right shift

```
i = 6;          /* 0000 0000 0000 0110 */
j = i << 2      /* 0000 0000 0001 1000 */
k = i >> 2      /* 0000 0000 0000 0001 */
```

- set bit at index 10 (start indexing at 0)

```
j = 10;
i = 1 << j      /* 0000 0100 0000 0000 */
```

Bitwise Complement, And, Or Xor

- ~ complement
- & and
- ^ xor
- | or

```
i = 17;    /* 0001 0001 */
j = 3;     /* 0000 0011 */
k = ~j;    /* 1111 1100 */
m = i & j  /* 0000 0001 */
n = i | j  /* 0001 0011 */
o = i ^ j  /* 0001 0010 */
```

Bitwise Complement, And, Or Xor

- ~ complement
- & and
- ^ xor
- | or

```
i = 69;      /* 0100 0101 */
j = 21;      /* 0001 0110 */
k = ~j;      /* 1110 1001 */
m = i & j    /* 0000 0100 */
n = i | j    /* 0101 0111 */
o = i ^ j    /* 0101 0011 */
```

Idioms

- Setting a bit string to all 1s:

```
i = ~0;
```

- Set all but the last 2 bits to 1:

```
i = ~0x3;
```

- Setting bit j

```
x = 1 << j;
or
x = 0;
x |= 1 << j;
```

Options, Masks, or Flags

- Flags are often implemented as a bit mask
- Example:

```
open(“temp”, O_WRONLY | O_CREAT);
#define O_RDONLY       00
#define O_WRONLY       01
#define O_RDWR         02
#define O_CREAT      0100
```

Watch out

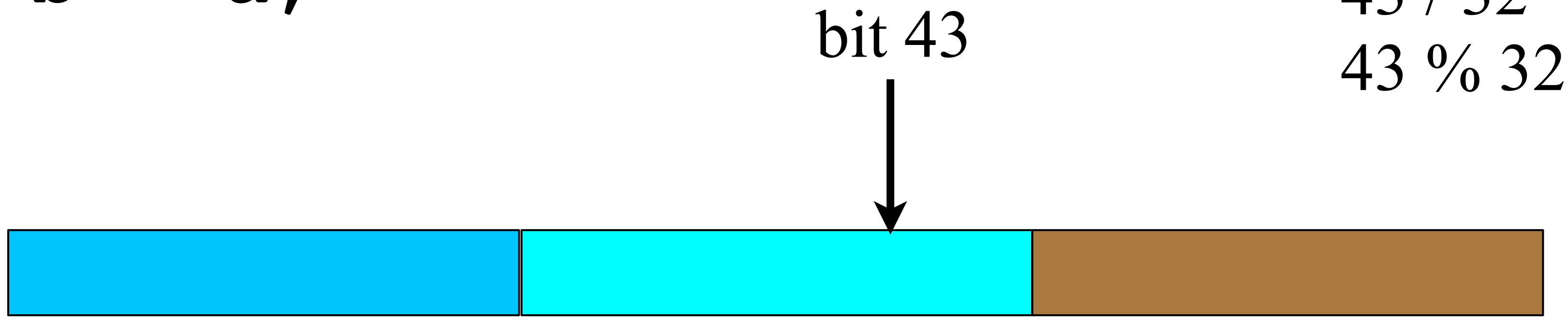
```
i = 2;                  /*0000 0010 */
j = 1;                  /*0000 0001 */

if( i & j)
  printf(“i and j = %d\n”, i & j);
if (i && j)
 printf(“both true %d\n”, i && j);
```

Arrays of bit strings

- FD_SETSIZE is bigger than 32.

```
struct bits {
  unsigned int field[N];
}
typedef struct bits Bitstring;
Bitstring a, b;
setzero(&a);
b = a;
```

43 / 32 = 1
43 % 32 = 11



Setting and unsetting

```
void set(unsigned int bit, Bitstring *b) {
  int index = bit / 32;
  b->field[index] |= 1 << (bit % 32);
}

void unset(unsigned int bit, Bitstring *b) {
  int index = bit / 32;
  b->field[index] &= ~(1 << (bit % 32));
}
```

Testing and emptying

```
int ifset(unsigned int bit, Bitstring *b) {
  int index = bit / 32;
  return ((1 << (bit % 32)) & b->field[index]);
}

int setzero(Bitstring *b){
  if(memset(b,0, sizeof(Bitstring)) == NULL)
    return 0;
  else
    return 1;
}
```

Printing

```
char *intToBinary(unsigned int number) {
  char *binaryString = malloc(32+1);
  int i;
  binaryString[32] = '\0';
  for (i = 31; i >= 0; i--) {
    binaryString[i] = ((number & 1) + '0');
    number = number >> 1;
  }
  return binaryString;
}
```