0 Contents

1 C

1.1 fixed size types

- int8_t, int16_t, int32_t, int64_t
  fixed integers with bit count
- intmax_t max size int on platform
- intptr_t signed integer with the size of an address on this platform
- uint8_t, uintptr_t unsigned versions
- size_t
  this is used in containers, the reson for this is that *this has the max size that for example an array can be.*
  This is unsigned!

1.2 Addition of pointers

If you try to add or subtract 2 pointers to get the amount of sizeof(t) difference, then you can only do this with the exact same type, something like signed int and unsigned int will not work!

```
int32_t *y = 100;
int32_t *x = 120;
ptrdiff_t z = x - y; // z == 5
uint32_t *u = 120;
ptrdiff_t p = u - y;  // Error: Different ptr types
```

1.3 Index Operator on Pointers

You can index on pointers like an array, this can be used to get elements on any object.
Note that you have to manually make sure to stay within the bounds of that object, as otherwise you will have *undefined behavior*.

```
int32_t x = 0;
int32_t *y = &x;
y[0] = 0x42;       // same as: x = 0x42;
(&x)[0] = 0x42;    // same
0[&x] = 0x42;      // same
100[200] = 0x42;   // Error: no address
```

1.4 Padding

When you mix and match different types of different sizes inside of a struct, then the compiler will include padding based on the bigger type:

```
struct {
char c;     // Offset 0
int32_t x;  // Offset 4 --> Padding
char d;     // Offset 8
} t;        // sizeof t == 12
# structure matters!!
struct {
char c;     // Offset 0
char d;     // Offset 1
int32_t x;  // Offset 2 --> Padding
} t;        // sizeof t == 6
```

2 Filesystems

2.1 Ext2

2.2 Ext4

3 Processmodels

4 Communication and Synchronization

5 Programs and libraries

6 Graphical Overlays