C - Basic Features & Pointers

CS238P: Principles of operating systems - Fall'18

University of California, Irvine

Data types and Control flow

- char (1 byte)
- int, long (4/8 bytes)
- float, double
- `if`
- `switch`
- `for`
- `while`
- the forgotten `do...while`

Structures

- Define a struct

  ```
  struct sandwish{
    int bread_size;
    char content;
    unsigned char taste;
  };
  ```

- Declare struct

  ```
  struct sandwish s0;
  ```

- Use Struct

  ```
  s0.bread_size = 4;
  ```

Hw1(xv6 shell)

- `if...else`

```
pid = fork();
if(pid == 0)
  printf("I am the child");
else if(pid == -1)
  perror("fork didnt work");
else{
  ... parent code;
}
```

- `switch...case`

```
switch(cmd->type){
case '>': ...; break;
default: ...; break;
}
```

- Functions
    - Process creation (`fork`, `exec`)
    - File I/O (`open`, `close`, `read`, `write`)

      ```
      fd = open(rcmd->file, rcmd->mode);
      ```
- Typecasting (next slide)
- Command line arguments (`argv`)

Typecasting

- Change the type of the object for a single operation

  ```
  someFunction((dest_type) source);
  ```

- Change the type of the object, and save it

  ```
  Dest_Type var = (dest_type) source;
  ```

- Pass generic objects

  ```
  struct cmd { int type; };
  struct execcmd {
    int type;
    char *argv[MAXARGS];
  };
  void runcmd(struct cmd *myArg) {
    switch(myArg->type){
      ...
      case ...:
        castedArg = (struct execcmd*)myArg;
    }
  }
  ```

Typecasting

- Pass generic objects

```
struct cmd { int type; };
struct execcmd {
  int type;
  char *argv[MAXARGS];
};

struct cmd* execcmd(void) {
  struct execcmd *result;
  ...
  return (struct cmd*)result;
}
```

- Beware of typecasting! (demo: ptr.c)

Arrays

- Collection of objects of the same data type
- Accessed by index (`0 ... size - 1`)
- String is an array of characters (demo: ptr.c)
- No reference operator

```
printf("Address of a \%p | \%p\n", a, &a);
>> Address of a 0x7aff07024060 | 0x7aff07024060
```