System-Level Programming

10 Variables

J. Kleinöder, D. Lohmann, V. Sieh, P. Wägemann

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Summer Term 2024

http://sys.cs.fau.de/lehre/ss24

Definition of Variables [≈Java]

- **variable** := container for values ($\mapsto$ memory space)
- Syntax (definition of variables):

  $sc_{opt}\ type_{opt}\ id_1\ [\ =\ expr_1]_{opt}\ [,\ id_2\ [\ =\ expr_2]_{opt}\ ,\ \ldots]_{opt}$;

  - sc_{opt} — storage class of the variable, `auto`, `static`, or none [≈Java]
  - *type* — type of the variable, `int` if no type is given ($\mapsto$ bad style!) [=Java] [≠Java]
  - id_i — name of the variable [=Java]
  - $expr_i$ — expression for initial value; if no value gets assigned, the content of non-`static` variables is **undefined** [≠Java]

Definition of Variables (continued)

- Variables can be defined at different positions
  - global — outside of any function, usually at the beginning of the file
  - local — at the begin of a `{ block }`, directly after the opening curly bracket — C89
  - local — everywhere where an expression is valid — C99

```
int a = 0;                      // a: global
int b = 47;                     // b: global

void main(void) {
  int a = b;                    // a: local to function, covers global a
  printf("%d", a);
  int c = 11;                   // c: local to function (C99 only!)
  for (int i=0; i<c; i++) {     // i: local to for-block (C99 only!)
    int a = i;                  // a: local to for-block,
  }                             //    covers function-local a
}
```

Definition of Variables (continued)

- Variables can be defined at different positions
  - global — outside of any function, usually at the beginning of the file
  - local — at the begin of a `{ block }`, directly after the opening curly bracket — C89
  - local — everywhere where an expression is valid — C99

```
int a = 0;                      // a: global
int b = 47;                     // b: global

void main(void) {
  int a = b;                    // a: local to function, covers global a
  printf("%d", a);
  int c = 11;                   // c: local to function (C99 only!)
  for (int i=0; i<c; i++) {     // i: local to for-block (C99 only!)
    int a = i;                  // a: local to for-block,
  }                             //    covers function-local a
}
```

We will have a closer look at global variables when talking about **modularisation** ↪ 12–5