Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 - WS 2022/23

Prof. Dr. Michael Rohs, Tim Dünte, M.Sc., Jan Feuchter, M.Sc.

Präsenzübung 12

Diese Aufgaben sind zur Lösung während der einstündigen Präsenzübung gedacht.

Aufgabe 1: sprintf, snprintf und scanf

a) Ist dieser Code in Ordnung? Begründen Sie. Was ist die Ausgabe?

```c
#include <stdio.h>
int main(void) {
    char *s = "1234567890";
    char a[4];
    sprintf(a, "%s", s);
    puts(a); // output string, followed by line break
    return 0;
}
```

b) Ist dieser Code in Ordnung? Begründen Sie. Was ist die Ausgabe?

```c
#include <stdio.h>
int main(void) {
    char *s = "1234567890";
    char a[4];
    snprintf(a, 4, "%s", s);
    puts(a); // output string, followed by line break
    return 0;
}
```

c) Ist dieser Code in Ordnung? Begründen Sie. Was ist die Ausgabe?

```c
#include <stdio.h>
int main(void) {
    char s[5];
    int matches = scanf("%s", s);
    printf("%d: %s\n", matches, s);
    return 0;
}
```

d) Falls in (c) ein Problem besteht, wie lässt es sich beheben?

e) Was ist die Ausgabe dieses Programms für die Eingabe "Oh, schneit es“?

```c
int main(void) {
    char a[60];
    int i = scanf("%20s%20s%20s", a, a + 20, a + 40);
    printf("%d: %s %s?\n", i, a + 40, a + 20);
    return 0;
}
```

Aufgabe 2: mehrere .c-Dateien

a) Es seien die Dateien a.c, a.h, b.h und c.h mit folgenden Inhalten gegeben:

a.c:
```c
#include "a.h"
#include "b.h"
int a(int i) { return i; }
int main(void) {
  return 0;
}
```

a.h:
```c
#include "c.h"
int a(int i);
```

b.h:
```c
#include "c.h"
double b(double f);
```

c.h:
```c
void c(int x);
```

Welches Problem liegt hier vor und wie lässt es sich lösen?

b) Es seien die Dateien a.c, b.h und b.c mit folgenden Inhalten gegeben:

a.c:
```c
#include <stdio.h>
#include "b.h"
int x = 123;
int main(void) {
    set_value(x);
    printf("%d\n", get_value());
    return 0;
}
```

b.h:
```c
void set_value(int v);
int get_value(void);
```

b.c:
```c
int x = 0;
void set_value(int v) { x = v; }
int get_value(void) { return x; }
```

Welches Problem liegt hier vor und wie lässt es sich lösen?

Aufgabe 3: Lifetime, Scope und Linkage

Überlegen Sie sich für jede Zeile in der Tabelle ein Codebeispiel.

Zeile	Level	Declaration/Definition	Keyword	Lifetime	Scope	Linkage	Public?
1	file	variable definition	static	program	rest of file	internal	no
2	file	variable declaration	extern	program	rest of file		a reference
3	file	variable definition or declaration	none	program	rest of file	external	yes (may be a reference)
4	file	function declaration or definition	static	program	rest of file	internal	no
5	file	function declaration or definition	extern (optional)	program	rest of file	external	yes
6	block	variable declaration	extern	program	rest of block		a reference
7	block	variable definition	static	program	rest of block	no linkage	no
8	block	variable definition	auto (optional)	block	rest of block	no linkage	no

Aufgabe 4 (optional): Makefile für separate Compilierung mehrerer .c-Dateien

Gegeben sei folgendes Makefile (mit Zeilennummern):

```
1. OBJECTS = a.o b.o          ← Variablendefinition
2. e.exe: $(OBJECTS)
3.     gcc $(OBJECTS) -L../lib -lprog1 -o $@
4. %.o: %.c
5.     gcc -c -iquote../lib $<
6. a.o: a.c b.h
7. b.o: b.c b.h
```

Zur Erinnerung: Eine Makefile-Regel hat die Grundform:

```
Ziel: Abhängigkeiten des Ziels
<TAB>Aktion
```

Das erste Ziel im Makefile ist das Default-Ziel, die verwendet wird, wenn `make` ohne Optionen aufgerufen wird.

In welcher Reihenfolge werden die Regeln bzw. Kommandos des obigen Makefiles bei Aufruf von `make` (ohne Parameter) ausgeführt, wenn bereits unmittelbar zuvor `make` aufgerufen und danach `b.h` editiert wurde? Verwenden Sie zur Erläuterung die angegebenen Zeilennummern.