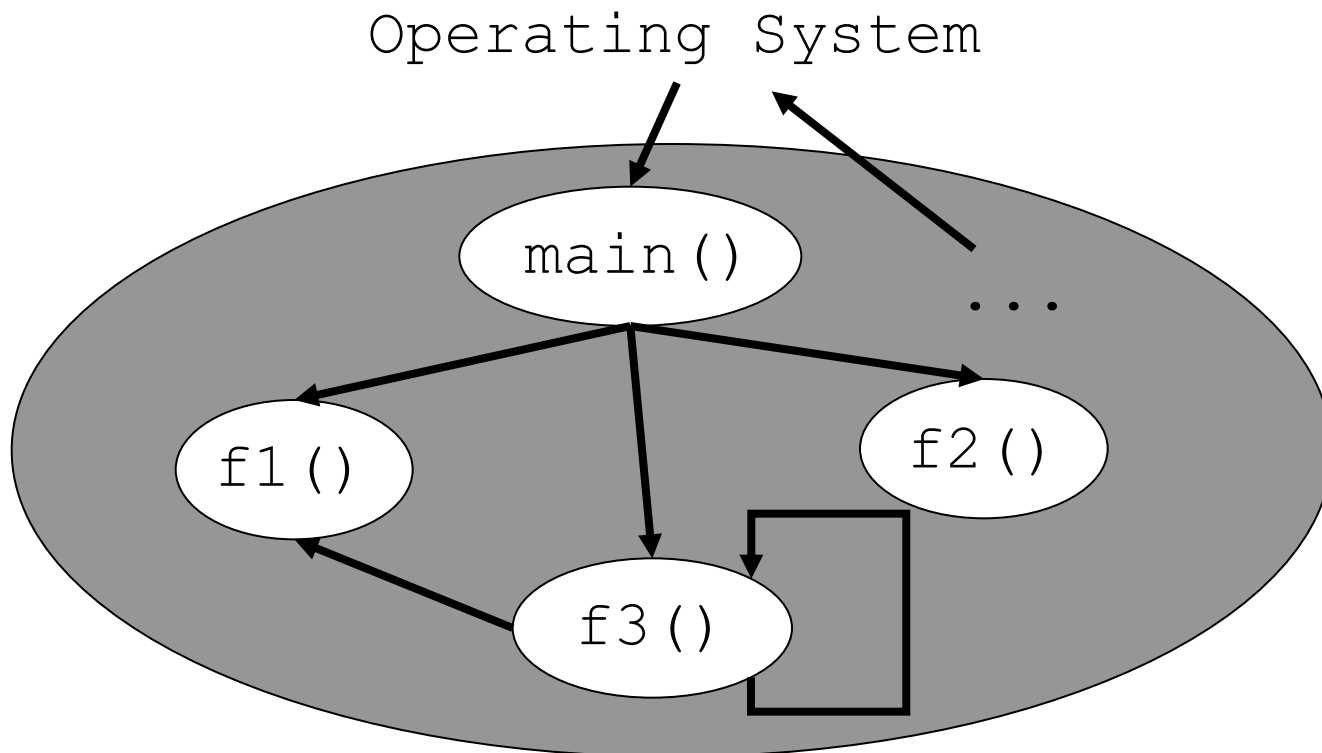


Computer Science 1

Introduction to Programming

Functions

➤ Global Picture



➤ Syntax:

```
✓ ReturnType Name (Parameter1, ... ParameterN) {  
    Statements;  
}
```

➤ Example:

```
✓ int main() {  
    return 0;  
}
```

➤ Function `power(x, n)` computes x^n

Return
Type

Function name

```
int power(int x, unsigned int n) {  
    int p=1;
```

```
    while(n>0) {  
        p = p * x;  
        n--;  
    }
```

```
    return p;  
}
```

Parameterlist
separated by commas

Returns result of
computation
(of type ReturnType)

Function call from main:

```
include <stdio.h>
```

Function prototype

```
int power(int x, unsigned int n);
```

```
void main()  
{
```

Function call
Assignment of
return value

```
    int result=power(2,3);  
    printf("2 power 3: %d\n",result);
```

```
}
```

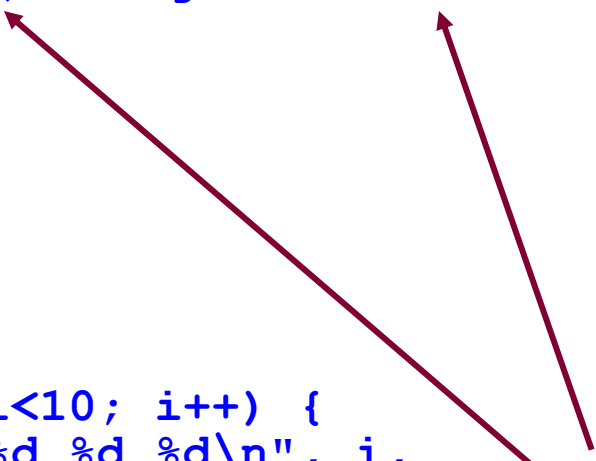
- Actual parameters
- Formal parameters

```
include <stdio.h>
```

```
int power(int x, unsigned int n)
{
    ...
}
```

```
void main() {
    int i;

    for (i=0; i<10; i++) {
        printf("%d %d %d\n", i,
                power(2,i),
                power(-3,i));
    }
}
```



The diagram consists of two red arrows. One arrow originates from the '2' in the first call to 'power(2,i)' and points to the 'int x' parameter in the 'power' function definition. The second arrow originates from the 'i' in the same call and points to the 'unsigned int n' parameter in the 'power' function definition. This illustrates how the actual parameters are passed to the formal parameters of the function.

```
int a=1; /* global variable */

void f() {
    int b=1;
    int c=2;
    printf("%d, %d, %d\n", a, b, c);

    a=a+1;
    b=b+1;
    c=c+2;
}

int main() {
    int c=1;
    while (a<4) {
        f();
        c++;
    }
}
```

Ausgabe:

1, 1, 2
2, 1, 2
3, 1, 2