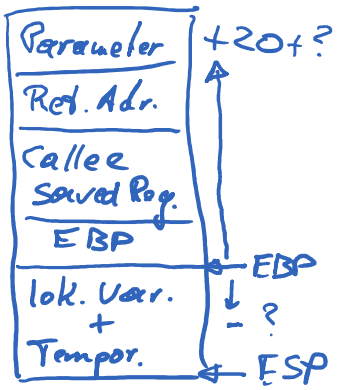
Dynamisch verwaltete lokale Umgebungen

Montag, 13. Mai 2019 07:41

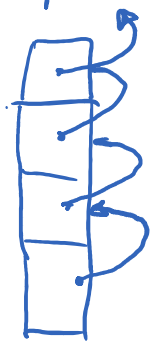
Prolog Stackframe Epilog

```
push
push
push
push EBP
mov EBP, ESP
sub ESP, #Bytes
```



Wozu: Bestandteil der C Calling Conventions, bzw. Laufzeitorganisation

Situation:



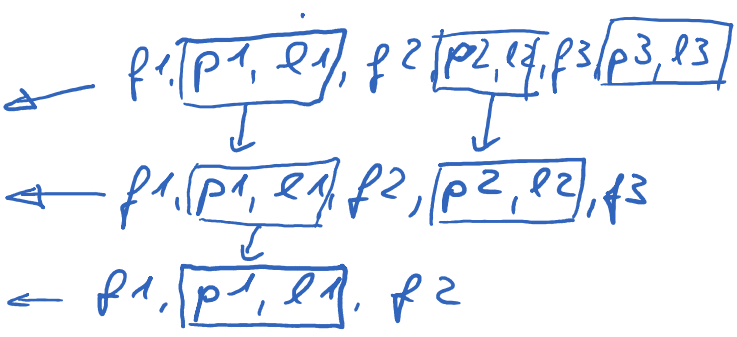
≙ rückwärts verkettete lineare Liste für C

Keine Funktionsdefinitionen in C, C++, Java...

Wechsel zu dynamisch, lokalen Umgebungen:
(in C-Syntax, geht in C nicht!)

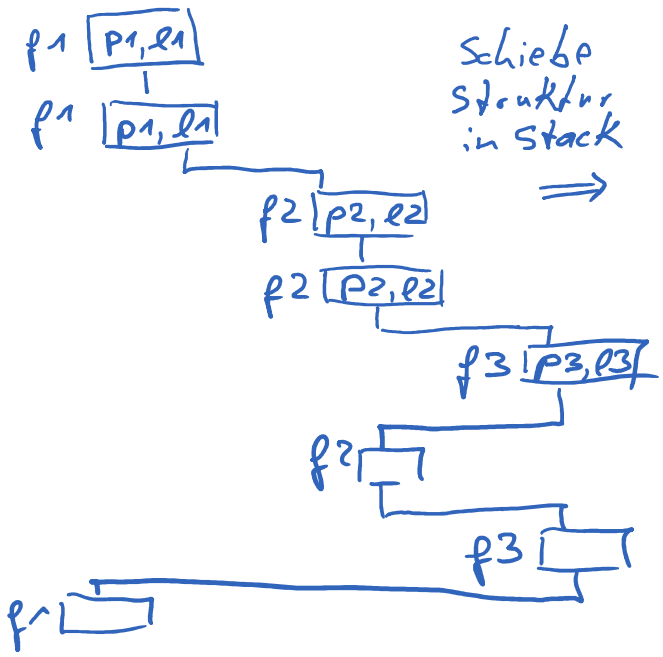
```
int f1 (int p1){
  int l1;
  int f2 (int p2) {
    int l2;
    int f3 (int p3){
      int l3
      code Rumpf f3
    }
    code Rumpf f2
  }
  code Rumpf f1
}
```

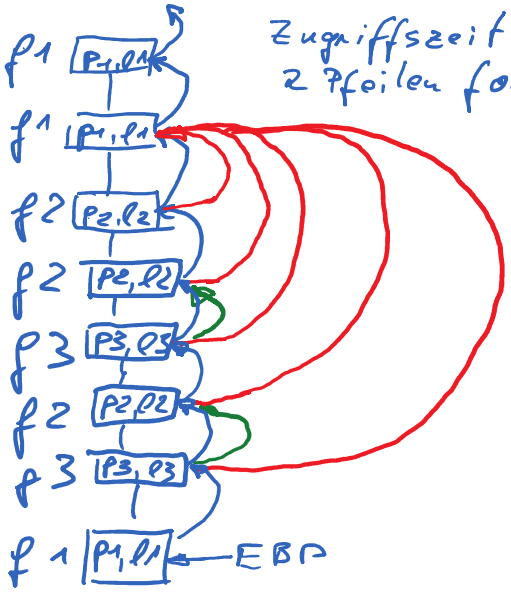
sichtbar:



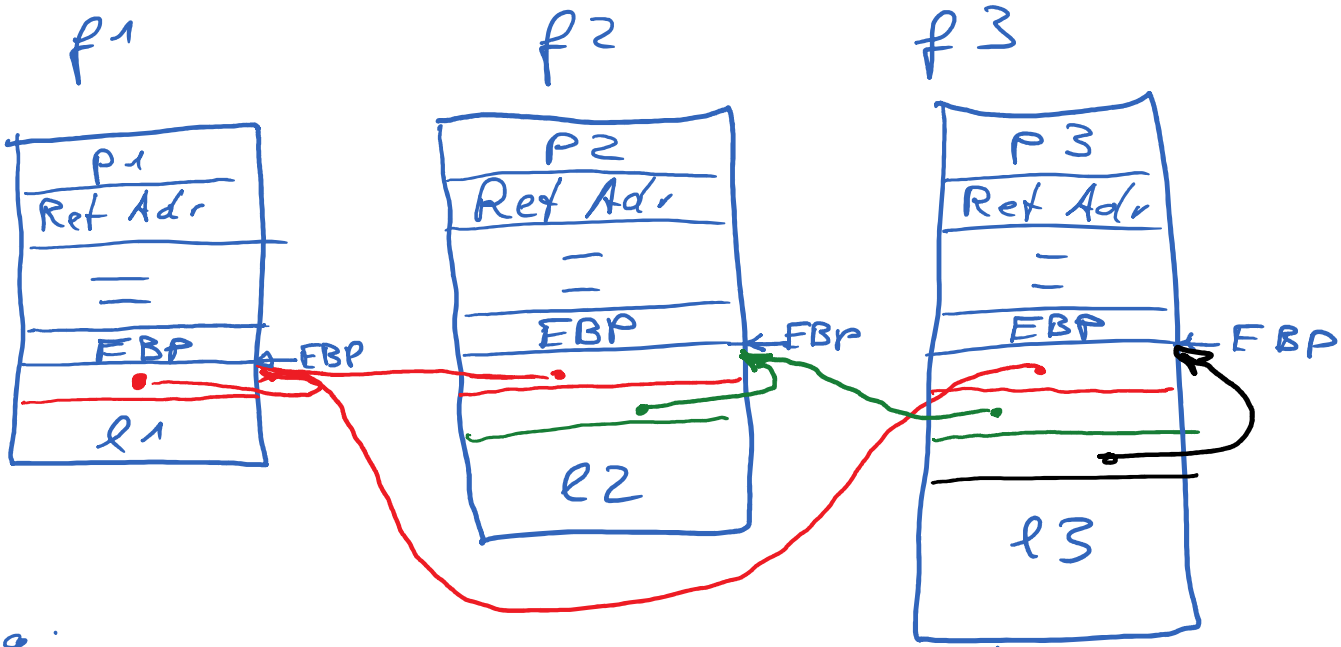
}

Laufzeitstruktur:





Aufbau Frames:



Prolog:

```
push EBX
push ESI
push EDI
Enter Sizeof(l1),1
```

```
push EBX
push ESI
push EDI
Enter Sizeof(l2),2
```

```
push EBX
push ESI
push EDI
Enter Sizeof(l3),3
```

Erklärung

Enter sizeof(l1), 1

wie Enter ..., 1 + kopiere 1

+ kopiere 2 Pointer

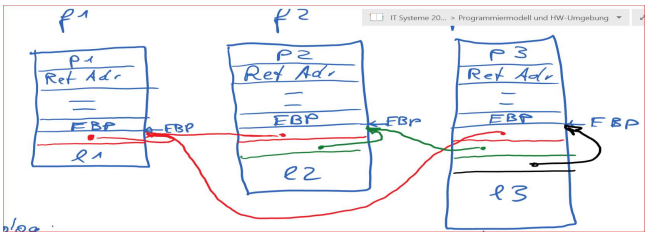
Enter sizeof(l1), 1 | + Kopiere 1 Pointer | + Kopiere 2 Pointer

≙

push EBP
mov EBP, ESP
push EBP // roter Pfeil
sub ESP, sizeof(l1)

Opcode	Instruction	Description
C8 *iw* 00	ENTER *imm16*,0	Create a stack frame for a procedure
C8 *iw* 01	ENTER *imm16*,1	Create a nested stack frame for a procedure
C8 *iw* ib	ENTER *imm16,imm8*	Create a nested stack frame for a procedure

Zugriff auf Parameter und lokale Variablen



f1

p1 DWORD PTR [EBP+20]

l1 DWORD PTR [EBP-8]

✓

f2

p2 →

l2 DWORD PTR [EBP-12]

p1, mov EBX, [EBP-4]
... DWORD PTR [EBX+20]

l1 analog

f3

p3 →

l3 ... -16

p1 →

l1 →

p2