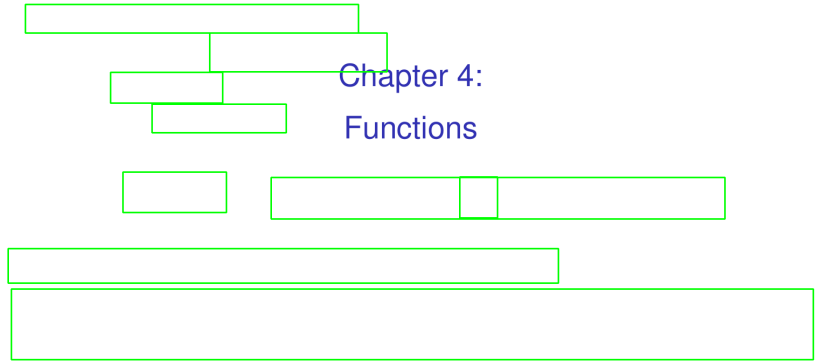
Script **generated by TTT**

Title: Petter: Compiler Construction (02.07.2020) - 56: Stack Frames

Date: Fri Jul 03 13:18:47 CEST 2020

Duration: 09:41 min

Pages: 5



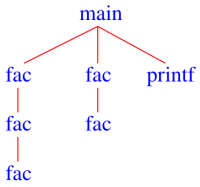
Chapter 4: Functions

Memory Management in Functions

```
int fac(int x) {
  if (x<=0) return 1;
  else return x*fac(x-1);
}
```

```
int main(void) {
  int n;
  n = fac(2) + fac(1);
  printf("%d", n);
}
```

At run-time several instances may be active, that is, the function has been called but has not yet returned.
The recursion tree in the example:



Memory Management in Function Variables

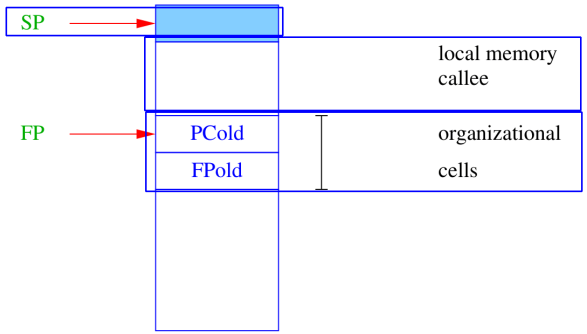
The formal parameters and the local variables of the various instances of a function must be kept separate

Idea for implementing functions:

- set up a region of memory each time it is called
- in sequential programs this memory region can be allocated on the stack
- thus, each instance of a function has its own region on the stack
- these regions are called stack frames

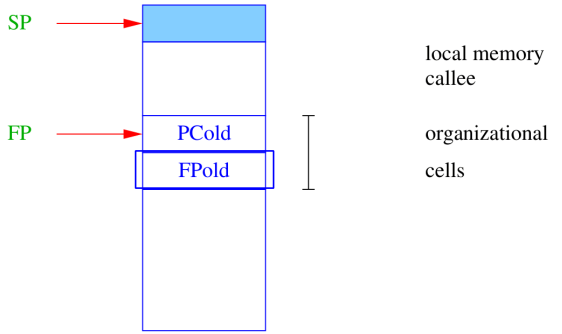
Organization of a Stack Frame

- stack representation: grows upwards
- SP points to the last used stack cell



Organization of a Stack Frame

- stack representation: grows upwards
- SP points to the last used stack cell



- FP $\widehat{=}$ frame pointer: points to the last organizational cell
- used to recover the previously active stack frame