MPW PQR4 Appendix C
Release Notes

Profiling Recursive Procedures

This Appendix illustrates in some detail the behavior of **Proff/PrintProff** when applied to a recursive function or procedure. Only direct recursion is analyzed; the treatment of indirect recursion is similar.

The function studied was the familiar recursive factorial:

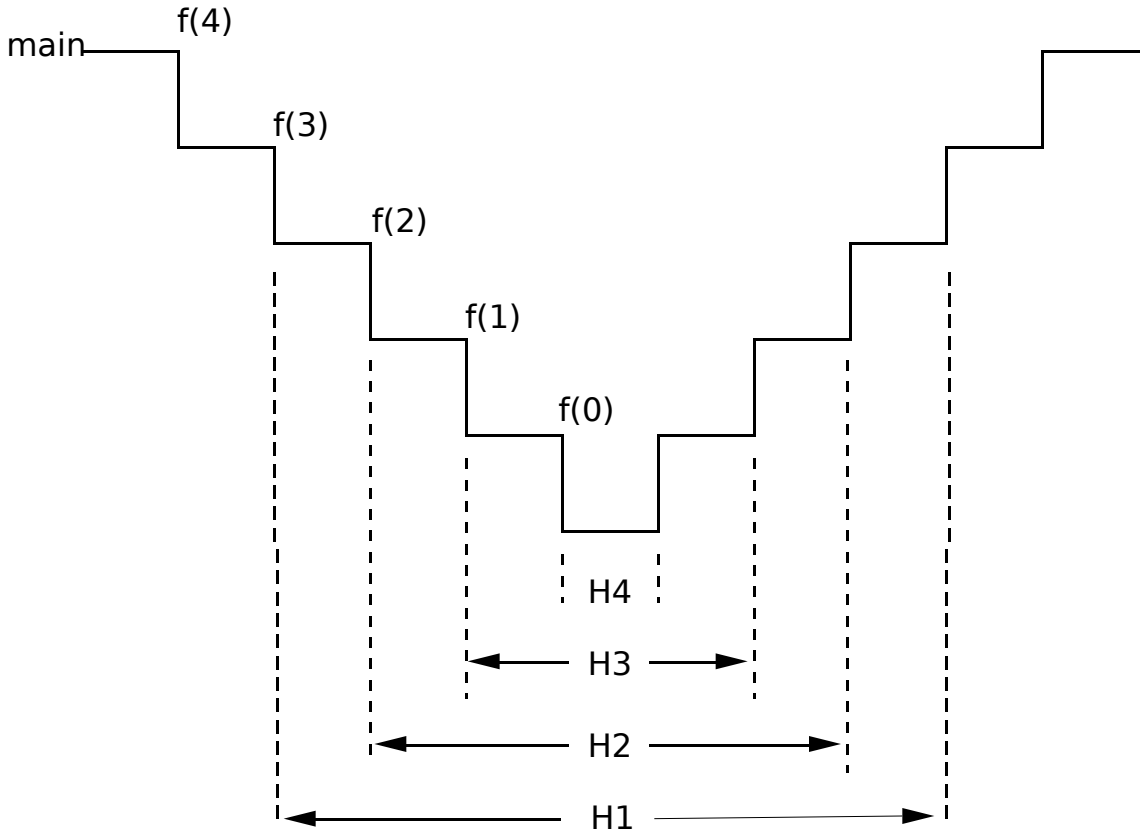
```
int f(int n)
{
      return(n == 0 ? 1 : n * f(n - 1);
}
```

This function, called from `main()`, was executed and profiled for `n` having values from 1 through 6. The results are shown in the table below. All times are in microseconds.

		Hierarchical Time	Flat Time
f(0)	called from `main`	22	22
	called from `f`	--	--
	total	22	22
f(1)	called from `main`	2737	2715
	called from `f`	22	22
	total	2760	2737
f(2)	called from `main`	5544	3104
	called from `f`	2462	2440
	total	8007	5544
f(3)	called from `main`	7562	2469
	called from `f`	7553	5092
	total	15115	7562
f(4)	called from `main`	10198	2639
	called from `f`	15108	7559
	total	25306	10198
f(5)	called from `main`	12809	2794
	called from `f`	25273	10014
	total	38082	12809
f(6)	called from `main`	15117	2416
	called from `f`	38001	12700
	total	53119	15117

A number of relationships can be seen immediately in the above data. It is clear that the time to calculate `f(0)` is 22 µsec. It is also clear that the difference between the hierarchical and flat times for `f(1)` is just the 22 µsec. for `f(0)`. Looking further, we see that in the calls from `main`, the flat times for `f(1)` and beyond are roughly constant and represent the execution time of the named recursion level plus the overheads (procedure call, exit, and monitoring) for the call of the next level. It can also be seen that the hierarchical times for the calls from `main` for `f(n)` are roughly equal to `n` times the constant flat time. But what are those ever growing times shown for the calls of `f` from `f`? (Note that the total times are just the respective sums of the times for calls from `main` and calls from `f`.)

The following diagram may help explain what is happening.



Time in the diagram increases from left to right. The first call of `f` from `f` is the call of `f(3)`. H1 is the hierarchical time of this call. Similarly, H2, H3, and H4 are the hierarchical times for the calls of `f(2)`, `f(1)`, and `f(0)`. The following relationships hold:

H1 = F1 + H2
H2 = F2 + H3
H3 = F3 + H4
H4 = F4

Now, substituting up the line in the above equations, we get:

H4 = F4
H3 = F3 + F4
H2 = F2 + F3 + F4
H1 = F1 + F2 + F3 + F4

The reported hierarchical time for `f` as called from `f` is H1 + H2 + H3 + H4, since all four of these calls are taken by the profiler as instances to be summed. Summing the above equations, we get this time to be F1 + 2*F2 + 3*F3 + 4*F4.

Now F4 is the flat time for calculating f(0), and F2, F3, and F4 are approximately equal, and are the flat times for the higher recursion levels individually. Renaming F4 as Fs (s for small), and the common value of the others as Fb (b for big), we find the time for f(4) to be given by 4*Fs + 6*Fb. Now Fs was 22 µsec. Fb can be estimated by averaging the six values of the flat time of `f` called from `main`. This average value is 2690 µsec. Applying the formula, the predicted value of the hierarchical time of `f` called from `f` for `f(4)` is 16228, compared to an observed value of 15108. The predicted value is about 7% high. A similar and consistent discrepancy was found when comparing the predicted to observed values for the other `n`-values. It is not known yet whether the discrepancy is due to an overlooked phenomenon, or is just an artifact caused by fluctuations in the "constant" flat time; the six measurements have a *standard deviation* of 249, which is about 10%.