

Pensieve header: A profiler for Mathematica.

```
Print["This is Profile.m of http://drorbn.net/AcademicPensieve/Projects/Profile/. This version: M
```

```
BeginPackage["Profile`"]
```

```
BeginProfile::usage = "BeginProfile[] begins a profiling session, with root name \
ProfileRoot. BeginProfile[root] does same with root name root."
```

```
ProfileRoot::usage = "The default root label of a profile."
```

```
EndProfile::usage = "EndProfile[] ends a profiling session and returns a ProfileData object conta
```

```
$Profiling::usage = "When $Profiling is True, profiling is on."
```

```
ProfileOn::usage = "ProfileOn[] turns profiling on."
```

```
ProfileOff::usage = "ProfileOff[] turns profiling off."
```

```
Profile::usage = "Profile[label,expr] evaluates expr while taking profiling \
data under the label label."
```

```
PP::usage = "PPlabel@expr is equivalent to Profile[label,expr]."
```

```
Called::usage = "n Called[parent,child] means that child was called n times \
by parent."
```

```
TimeUnder::usage = "t TimeUnder[parent,child] means that child spent time t \
under parent."
```

```
ProfileData::usage = "The result of EndProfile. Has format \
ProfileData[root,calls,total,self], where root is the profile root\
label, calls is a linear combination of tags with head Called, and\
total and self are linear combinations of tags with head TimeUnder."
```

```
PrintProfile::usage = "PrintProfile[root] prints the current profiling data under root \
(defaults to the current profile)."
```

```
Begin["`private`"]
```

```
ProfileOff[] := (
  $Profiling = False;
  Attributes[Profile] = {HoldFirst};
  Profile[Label_, expr_] := expr
);
ProfileOff[]
```

```

ProfileOn[] := (
  $Profiling=True;
  Attributes[Profile]={HoldRest};
  Profile[Label_,expr_] := (
    Block[
      {
        PreviousLabel=CurrentLabel,
        CurrentLabel=Label,
        EntryTime=TimeUsed[],
        AdjustedEntryTime
      },
      AdjustedEntryTime = EntryTime;
      CallingHistory += Called[PreviousLabel,CurrentLabel];
      value=expr;
      TotalTime += (TimeSpent=(TimeReading=TimeUsed[])-EntryTime) *
        TimeUnder[PreviousLabel,CurrentLabel];
      SelfTime += (TimeReading-AdjustedEntryTime) *
        TimeUnder[PreviousLabel,CurrentLabel];
    ];
    AdjustedEntryTime += TimeSpent;
    value
  )
)

```

```

PP_Label_ := Function[expr, Profile[Label, expr], {HoldAll}]

```

```

BeginProfile[] := BeginProfile[ProfileRoot]
BeginProfile[root_] := (
  ProfileOn[];
  CallingHistory = TotalTime = SelfTime = 0;
  RootEntryTime = AdjustedEntryTime = TimeUsed[];
  $ProfileRoot = CurrentLabel = root
)

```

```

EndProfile[] := (
  ProfileOff[];
  ProfileData[$ProfileRoot,CallingHistory,TotalTime,SelfTime]
)

```

```

$CurrentProfile := ProfileData[$ProfileRoot,CallingHistory,TotalTime,SelfTime]

```

```

TimeIn[Label_] := TimeIn[Label,$CurrentProfile]
TimeIn[Label_,ProfileData[_,_,_st_]] := st /. {
  TimeUnder[_ ,Label] -> 1,
  TimeUnder[_ ,_] -> 0
}

```

```

ProfileLabels[] := ProfileLabels[$CurrentProfile]
ProfileLabels[pd_ProfileData] := Block[
  {out={}},
  pd[[2]] /. Called[lbls_] := (out=Union[out,{lbls}]);
  Reverse[Last /@ Sort[{TimeIn[#,pd],#}& /@ out]]
]

```

```

PrintProfile[pd_ProfileData] :=
  StringTrim@StringJoin[PrintProfile[#1, pd] & /@ ProfileLabels[pd]];
PrintProfile[] := PrintProfile[$CurrentProfile];
PrintProfile[Label_] := PrintProfile[Label, $CurrentProfile];
PrintProfile[Label_, ProfileData[pr_, ch_, tt_, st_]] :=
  Module[{labelist = {}, i, l1, l2, out = {}, StreamOut},
    StreamOut[args___] := AppendTo[out, StringJoin[ToString /@ {args}] <> "\n"];
    l1 = Floor[N[Log[10, ch /. _Called -> 1]]];
    l2 = 3 + Floor[N[Log[10, tt /. _TimeUnder -> 1]]];
    StreamOut[
      Label, ": called ",
      ch /. {Called[_ , Label] -> 1, Called[_ , _] -> 0}, " times, time in ",
      TimeIn[Label], "/", tt /. {TimeUnder[_ , Label] -> 1, TimeUnder[_ , _] -> 0}
    ];
    ch /. Called[lbl_, Label] := (labelist = Union[labelist, {lbl}]);
    If[Length[labelist] > 0,
      StreamOut[" Parents:"];
      Do[
        StreamOut[StringForm["      (`) `~/` under `",
          PaddedForm[Coefficient[ch, Called[labelist[[i]], Label]], l1],
          PaddedForm[Coefficient[st, TimeUnder[labelist[[i]], Label]], {l2, 3}],
          PaddedForm[Coefficient[tt, TimeUnder[labelist[[i]], Label]], {l2, 3}],
          labelist[[i]]
        ]],
        {i, 1, Length[labelist]}
      ];
    ];
    labelist = {};
    ch /. Called[Label, lbl_] := (labelist = Union[labelist, {lbl}]);
    If[Length[labelist] > 0,
      StreamOut[" Children:"];
      Do[
        StreamOut[StringForm["      (`) `~/` above `",
          PaddedForm[Coefficient[ch, Called[Label, labelist[[i]]]], l1],
          PaddedForm[Coefficient[st, TimeUnder[Label, labelist[[i]]]], {l2, 3}],
          PaddedForm[Coefficient[tt, TimeUnder[Label, labelist[[i]]]], {l2, 3}],
          labelist[[i]]
        ]],
        {i, 1, Length[labelist]}}];
    StringJoin[out]
  ]

```

```
End[]; EndPackage[];
```

```
StreamOut[args___] := AppendTo[out, StringJoin@@(ToString /@ {args}) <> "\n"];
```

```
StreamOut[1, 2, 3]
```

AppendTo: out is not a variable with a value, so its value cannot be changed. 

```
AppendTo[out, 123  
]
```

? StringJoin

"s₁" <> "s₂" <> ..., StringJoin["s₁", "s₂", ...], or StringJoin[{"s₁", "s₂", ...}] yields a string consisting of a concatenation of the s_i. >>