TO: Distribution  
FROM: Ross E. Klinger  
DATE: December 3, 1973  
SUBJECT: Proposed new commands

Attached are the HPM write-ups and sample output for two new commands: snoop (sn), and print_snoop_data (psd). These commands provide the means by which the segment references of large, multi-procedure processes can be metered. They are an aid in determining the flow of execution within such a process, and more important, in which procedures a process is spending its time. The systems programmer and subsystem designer ought to benefit especially from locating the “hotspots” in a process. In particular, the Initializer and IO Daemons are likely candidates for “snooping”.

The snoop and psd commands augment the tracing facilities provided by the commands trace and page_trace, and the PL/I compile option -profile. The latter provides too fine a trace resolution for effective per process metering; trace is more of a debugging than metering tool; and page_trace cannot directly, or completely, meter the actual flow of a process.

1) Segment references are detected by snoop irrespective of page or segment faults. References to wired pages, or to pages which are largely core-resident due to frequent usage, can therefore be detected.

2) snoop uses a virtual CPU timer to control its sampling, so that actual non-ring 0 execution time is metered.

3) snoop provides for up to 32,767 samples, or approximately 9 hours of non-ring 0 CPU time at a sample rate of 1 second. page_trace is limited to approximately 350 samples.

The following are planned additions to print_snoop_data:

-brief to suppress the detailed trace.
-ignore s1,s2,... to exclude the specified segment numbers s1, s2, etc. when generating the histograms. Segments such as pli_operators are so frequently referenced that they tend to unbalance the histogram resolution.

---------------------------------------------------------------
Multics Project internal working documentation. Not to be reproduced or distributed outside the Multics Project.
A more extensive modification could eventually provide finer details of a segment reference, such as which page of a segment was referenced. snoop can currently resolve down to an offset within a component of a bound segment.

If you wish to try these commands, they currently reside in >udd>pdo>Klinger>public. In any case, I welcome your comments and suggestions. Send written comments to:

Ross E. Klinger
MIT Room 39-464

or mail comments to:

Klinger.PDO
Name: snoop, sn

The snoop command periodically samples the machine registers in order to determine which segments a process is referencing. Three output segments are produced, which are interpretable by the print_snoop_data command. (See the MPM write-up of the print_snoop_data command.)

Usage snoop -control_group-

Control arguments may be chosen from one of the following two control groups:

1) arguments which initiate sampling

-time n
-tm n

specifies the rate in milliseconds at which the process will be sampled. n must be a positive integer. The default is n = 1000; i.e., the process will be sampled once every second.

-segment name
-sm name

specifies the names to be given the three output segments. name may be either an absolute or relative pathname. name may end with the suffix .snoop; if it does not, .snoop will be appended. The output segments will be named as follows:

(entry portion of) name.snoop
(entry portion of) name.snoopp
(entry portion of) name.snoope

If name is an absolute pathname, the output segments will be placed in the appropriate directory. If name is a relative pathname, the output segments will be placed in the user's working directory. The default will cause the output segments to be placed in the user's working directory, with entry names as follows:

mm/dd/yy__hhmm.m_zzz_w~~osnoop
mm/dd/yy__hhmm.m_zzz_www.snoopp
mm/dd/yy__hhmm.m_zzz_www.snoope

Copyright 1973, Massachusetts Institute of Technology and Honeywell Information Systems Inc.
(See the MPM write-up of the date_time subroutine for an explanation of the above code.)

2) the argument which terminates sampling

-rese:
-specifies that the process is no longer to be sampled.

Notes

The machine registers can be sampled only when the process is running in a ring other than ring 0. Were a process to use, for example, a total of 100 seconds of processor time, and snoop, running at a sample rate of q = 1000, were to record only 23 samples, it would indicate that 77 seconds of processor time were spent in ring 0.

Under certain conditions, the contents of one of the machine registers sampled - the Temporary Segment Register (TSR) - may be invalid. This invalidity will be noted, but does not necessarily indicate that the process is in error.

At the maximum sample rate, 1 millisecond, execution time may be increased by as much as 50%. Using a 1 second sample rate, the increase in execution time is negligible.

If the process being sampled should be terminated without an invocation of snoop with the -reset option, interpretable output segments will still be produced; however, both the off-time and the last recorded sample will be invalid.
The `print_snoop_data` command interprets the three data segments produced by the `snoop` command, and produces a printable output segment which contains the following information: a detailed trace of segment references; a segment number to pathname dictionary; and histograms of the Procedure Segment Register (PSR) and Temporary Segment Register (TSR) segment reference distributions. (See the MPM write-up of the `snoop` command.)

**Usage**

`print_snoop_data name`

- `name` specifies the names of the data segments to be interpreted, as well as the name of the output segment to be produced. `name` may be either an absolute or relative pathname.
- `name` may end with the suffix `.snoop`: If it does not, `.snoop` will be appended.

If `name` is an absolute pathname, the specified directory will be searched for three segments with entry names as follows:

(entry portion of) `name.snoop`
(entry portion of) `name.snoopx`
(entry portion of) `name.snoope`

The output segment will be placed in the same directory with the entry name:

(entry portion of) `name.snoop`

If `name` is a relative pathname, the user's working directory will be searched for the data segments, and the output segment will be placed in the working directory, named as above.

**Notes**

1) `print_snoop_data` is able to detect a re-used segment number. The appearance of a parenthesized integer preceding a segment number indicates such a re-usage.

---

Copyright 1973, Massachusetts Institute of Technology and Honeywell Information Systems Inc.
The appearance of three such lines in the detailed trace would imply the following:

a) A reference was made to location 6542 in `bound_alpha_'. The particular component of `bound_alpha_' being referenced could not be determined. `bound_alpha_' was assigned segment number 234.

b) A reference was made to location 512 in `max35'. `max35' is a component of a bound segment whose name can be determined from the segment number to pathname dictionary. The segment `bound_alpha_' has been terminated, and when the segment of which `max35' is a component was initiated, it was assigned segment number 234.

c) A reference was made to location 6160 in `assign_'. The segment of which `max35' is a component has been terminated, and when `assign_ was initiated, it was assigned segment number 234.

2) The appearance of a segment number suffix (i.e., .1, .2, etc.) indicates a component of a bound segment.

```plaintext
310   >system_library_standard>bound_tl_term_
310.1  tssl_
310.2  translator_info_
```

The appearance of the above lines in the segment number to pathname dictionary would indicate that `tssl_ was the first component of `bound_tl_term_' to be referenced, and that `translator_info_ was the second component of `bound_tl_term_' to be referenced.

© Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.
1223 PSR 23013164
1224 PSR (1) 22412774
1225 PSR 23013204
1226 PSR 23013206
1227 PSR (2) 31116642
1228 PSR (2) 31117151
1229 PSR 23013206
1230 PSR (2) 31117151
1231 PSR 23013204
1232 PSR 25316156
1233 PSR 25316222
1234 PSR 23013204
1235 PSR 2531112
1236 PSR 22417076
1237 PSR 25311054

>system_library_1>bound_sss_wired_13204
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13206
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13206
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13204
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13204
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13204
>system_library_1>bound_sss_wired_13203
>system_library_1>bound_sss_wired_13204
>system_library_1>bound_sss_wired_13203