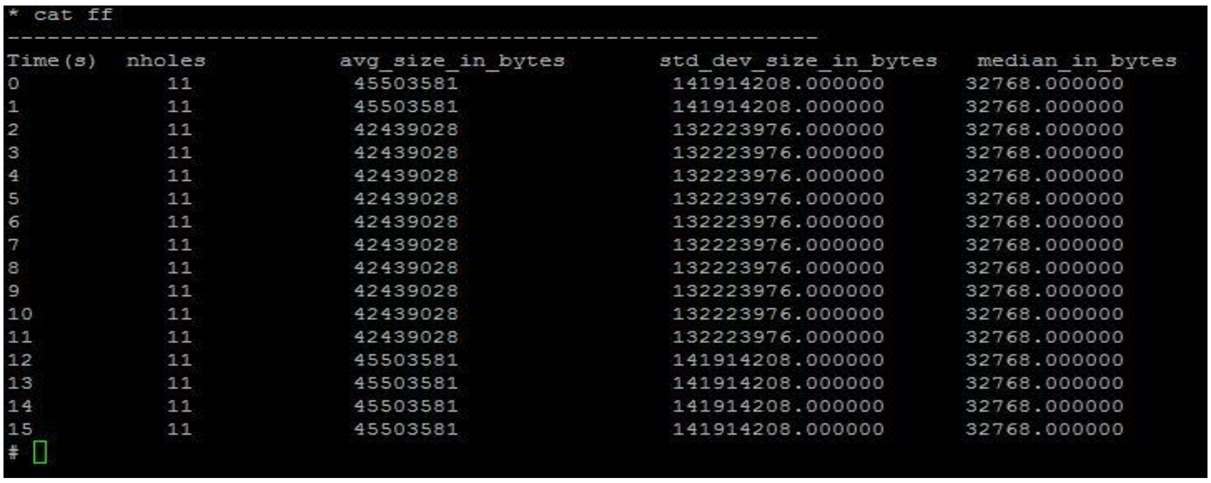
P5A Evaluation:

There are a total of 204 memory holes maintained by PM. This is defined in usr\include\minix\sys_config.h

#define _NR_PROCS 100
#define _NR_HOLES (2*_NR_PROCS+4)

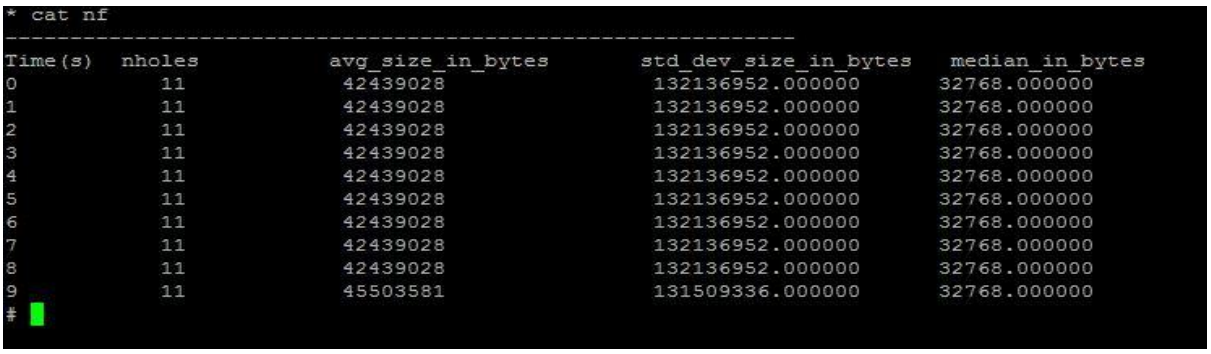
memlog.c – used to gather statistics regarding the number and the size of the used holes - if(pmi.pmi_holes[h].h_base && pmi.pmi_holes[h].h_len). This information is gathered once per second and the number of holes, their average size, the standard deviation of their size, and the median of their size are computed.

First fit: Holes with average size is used all throughout when compared to the other methods shown below. Though the average size of the holes increase on load and then decrease, the number of holes themselves does not change. In terms of performance since the algorithm looks for the first available hole which accommodates the requested size/clicks and therefore the runtime will be faster. However if there are holes which better fit the request further in the linked list, it may not be used. So this algorithm may not be efficient in memory utilization.



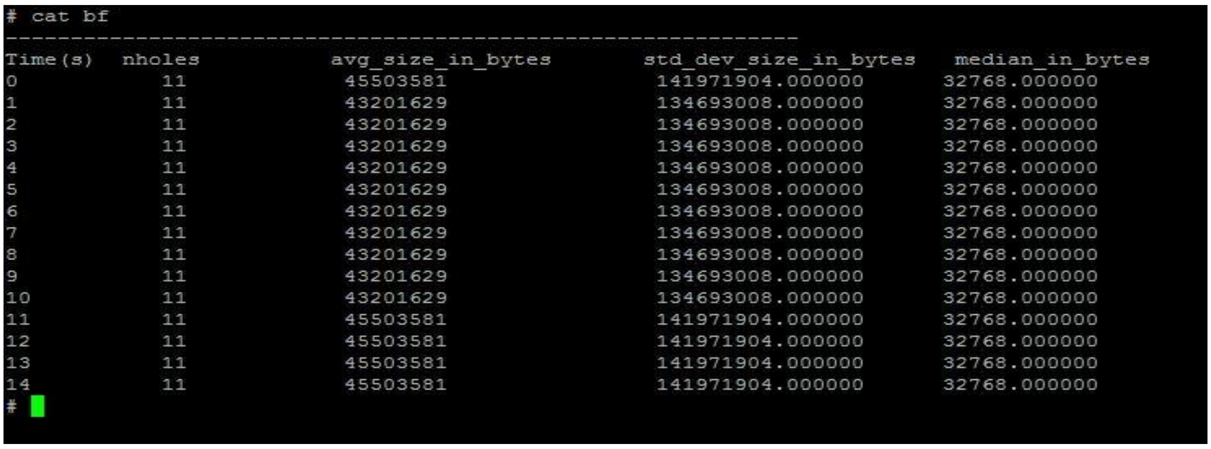
```
* cat ff
-------------------------------------------------------------
Time(s)  nholes          avg_size_in_bytes      std_dev_size_in_bytes   median_in_bytes
0           11            45503581               141914208.000000       32768.000000
1           11            45503581               141914208.000000       32768.000000
2           11            42439028               132223976.000000       32768.000000
3           11            42439028               132223976.000000       32768.000000
4           11            42439028               132223976.000000       32768.000000
5           11            42439028               132223976.000000       32768.000000
6           11            42439028               132223976.000000       32768.000000
7           11            42439028               132223976.000000       32768.000000
8           11            42439028               132223976.000000       32768.000000
9           11            42439028               132223976.000000       32768.000000
10          11            42439028               132223976.000000       32768.000000
11          11            42439028               132223976.000000       32768.000000
12          11            45503581               141914208.000000       32768.000000
13          11            45503581               141914208.000000       32768.000000
14          11            45503581               141914208.000000       32768.000000
15          11            45503581               141914208.000000       32768.000000
# 
```

Next fit: similar to first fit except the search starts from the last hole. It keeps track of where it is whenever it finds a suitable hole. The next time it is called to find a hole, it starts searching the list from the place where it left off last time, instead of always at the beginning, as first fit does. The average size varies. It is fast in terms of time but is not efficient with memory management.



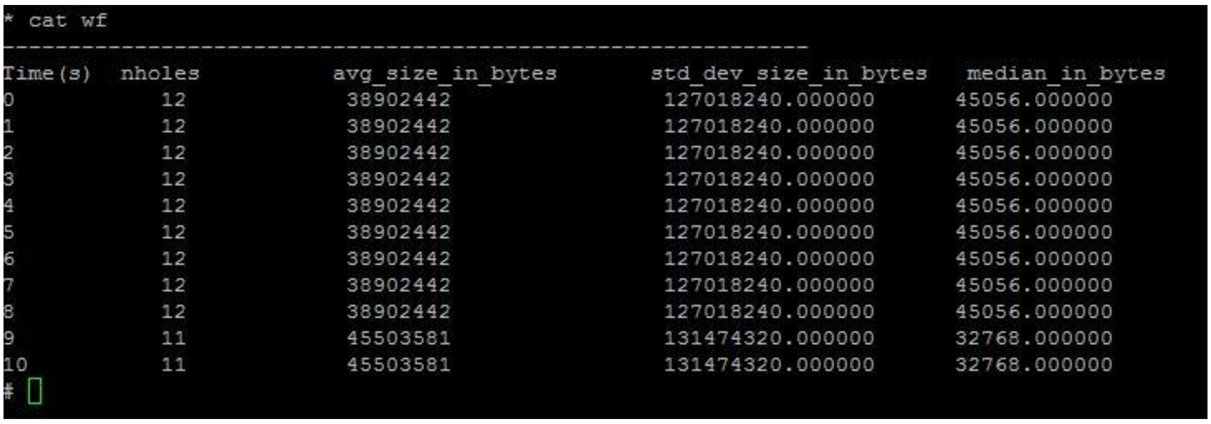
```
* cat nf
-------------------------------------------------------------
Time(s)  nholes          avg_size_in_bytes      std_dev_size_in_bytes   median_in_bytes
0           11            42439028               132136952.000000       32768.000000
1           11            42439028               132136952.000000       32768.000000
2           11            42439028               132136952.000000       32768.000000
3           11            42439028               132136952.000000       32768.000000
4           11            42439028               132136952.000000       32768.000000
5           11            42439028               132136952.000000       32768.000000
6           11            42439028               132136952.000000       32768.000000
7           11            42439028               132136952.000000       32768.000000
8           11            42439028               132136952.000000       32768.000000
9           11            45503581               131509336.000000       32768.000000
# 
```

Best fit: searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed. This can be seen in the logger table below. It is not as fast as first fit in terms of time but is efficient with memory management.



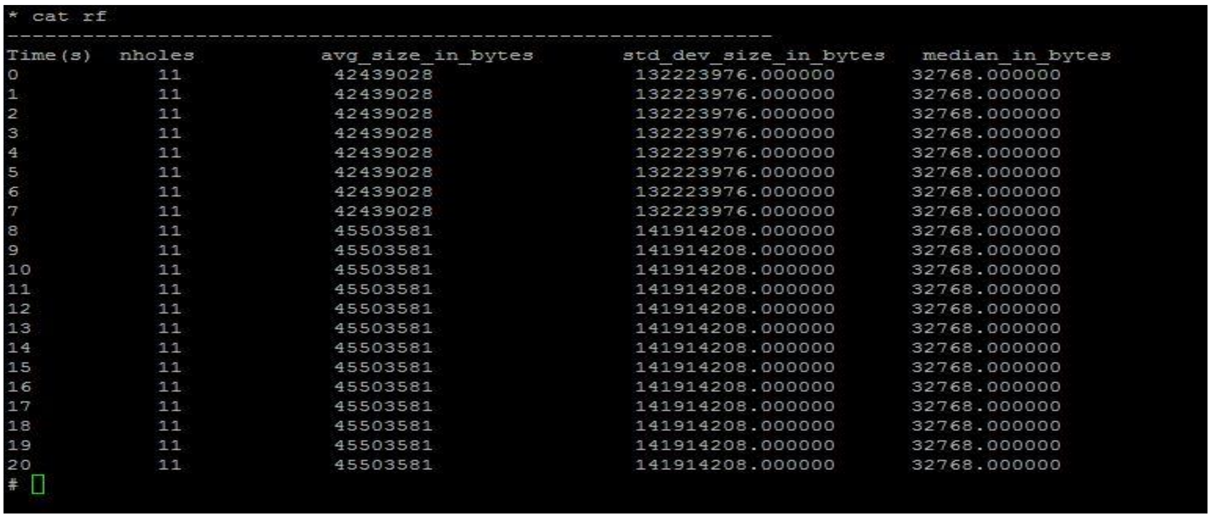
```
# cat bf
-----------------------------------------------------------------
Time(s)   nholes          avg_size_in_bytes       std_dev_size_in_bytes    median_in_bytes
0            11             45503581                141971904.000000        32768.000000
1            11             43201629                134693008.000000        32768.000000
2            11             43201629                134693008.000000        32768.000000
3            11             43201629                134693008.000000        32768.000000
4            11             43201629                134693008.000000        32768.000000
5            11             43201629                134693008.000000        32768.000000
6            11             43201629                134693008.000000        32768.000000
7            11             43201629                134693008.000000        32768.000000
8            11             43201629                134693008.000000        32768.000000
9            11             43201629                134693008.000000        32768.000000
10           11             43201629                134693008.000000        32768.000000
11           11             45503581                141971904.000000        32768.000000
12           11             45503581                141971904.000000        32768.000000
13           11             45503581                141971904.000000        32768.000000
14           11             45503581                141971904.000000        32768.000000
#
```

Worst fit: Has comparatively more no of holes when compared to the other allocation mechanisms and the average size of the holes here is much smaller. This policy allocates the process to the largest available free block of memory. This leads to elimination of all large blocks of memory, thus requests of processes for large memory cannot be met eventually. It is not efficient both in terms of time and memory management.



```
* cat wf
-----------------------------------------------------------------
Time(s)   nholes          avg_size_in_bytes       std_dev_size_in_bytes    median_in_bytes
0            12             38902442                127018240.000000        45056.000000
1            12             38902442                127018240.000000        45056.000000
2            12             38902442                127018240.000000        45056.000000
3            12             38902442                127018240.000000        45056.000000
4            12             38902442                127018240.000000        45056.000000
5            12             38902442                127018240.000000        45056.000000
6            12             38902442                127018240.000000        45056.000000
7            12             38902442                127018240.000000        45056.000000
8            12             38902442                127018240.000000        45056.000000
9            11             45503581                131474320.000000        32768.000000
10           11             45503581                131474320.000000        32768.000000
#
```

Random fit: checks for the fitting holes first and then randomly allocate holes.



```
* cat rf
-----------------------------------------------------------------
Time(s)   nholes          avg_size_in_bytes       std_dev_size_in_bytes    median_in_bytes
0            11             42439028                132223976.000000        32768.000000
1            11             42439028                132223976.000000        32768.000000
2            11             42439028                132223976.000000        32768.000000
3            11             42439028                132223976.000000        32768.000000
4            11             42439028                132223976.000000        32768.000000
5            11             42439028                132223976.000000        32768.000000
6            11             42439028                132223976.000000        32768.000000
7            11             42439028                132223976.000000        32768.000000
8            11             45503581                141914208.000000        32768.000000
9            11             45503581                141914208.000000        32768.000000
10           11             45503581                141914208.000000        32768.000000
11           11             45503581                141914208.000000        32768.000000
12           11             45503581                141914208.000000        32768.000000
13           11             45503581                141914208.000000        32768.000000
14           11             45503581                141914208.000000        32768.000000
15           11             45503581                141914208.000000        32768.000000
16           11             45503581                141914208.000000        32768.000000
17           11             45503581                141914208.000000        32768.000000
18           11             45503581                141914208.000000        32768.000000
19           11             45503581                141914208.000000        32768.000000
20           11             45503581                141914208.000000        32768.000000
#
```

The no of holes used in each method is 11 with varying sizes except for the worst fit where it is 12 smaller sized holes.