CIS 452

Lab 7 Report

Ashley Hendrickson

Muna Gigowski

Fall 2019

System Resource Limitations

|  |  |  |  |
| --- | --- | --- | --- |
| **System Object** | **Method** | **Value** | **Details** |
| Maximum # of semaphores per process | static | 256 | Used laptop; looked at limits.h in the Linux documentation |
| Maximum value of a (counting) semaphore | static | 32,767 | Used laptop; looked at limits.h in the Linux documentation |
| Maximum value of a (counting) semaphore | empirical | Hint : can use semct()l to set your own counting value for semaphore - play with that value until it errors out |  |
| Maximum size of a shared memory segment (bytes) | empirical |  |  |
| Page size (bytes) | dynamic | 4096 bytes | Used EOS Linux machine; created small program:  int main (int argc, char \*argv[])  {     printf("%ld\n", sysconf(\_SC\_PAGESIZE));     return 0;  } |
| Physical pages in system | dynamic | 4071585 | Used EOS Linux machine; created small program:  int main (int argc, char \*argv[])  {     printf("%ld\n", sysconf(\_SC\_PHYS\_PAGES));     return 0;  } |
| Maximum # of processes per user | dynamic | 63560 | Used EOS Linux machine; created small program:  int main (int argc, char \*argv[])  {     printf("%ld\n", sysconf(\_SC\_CHILD\_MAX));     return 0;  } |
| Maximum filesize (bytes) | dynamic | -1 | Used EOS Linux machine;created small program: |
| Maximum # of open files: hard limit | dynamic | 524288 | Used EOS Linux machine; created small program:  int main (int argc, char \*argv[])  {     struct rlimit rlim;     getrlimit(RLIMIT\_NOFILE, &rlim);     //printf("%ld\n", );     printf("%lld\n", (long long) rlim.rlim\_cur);     printf("%lld\n", (long long) rlim.rlim\_max);     return 0;  } |
| Maximum # of open files: soft limit | dynamic | 1024 | Used EOS Linux machine; created small program:  int main (int argc, char \*argv[])  {     struct rlimit rlim;     getrlimit(RLIMIT\_NOFILE, &rlim);     //printf("%ld\n", );     printf("%lld\n", (long long) rlim.rlim\_cur);     printf("%lld\n", (long long) rlim.rlim\_max);     return 0;  } |
| Clock resolution (msec) | dynamic | 100 ticks per second  Conversion:  100t/1s  1s/1000ms  = .1t/ms  = .9ms between ticks | Used EOS Linux machine; created small program:  int main (int argc, char \*argv[])  {     printf("%ld\n", sysconf(\_SC\_CLK\_TCK));     return 0;  } |