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	25,266,659,328	See code provided below for our solution to this question.
Maximum size of a shared memory segment (bytes)	empirical	32,767	See code provided below for our solution to this question.

Page size (bytes)	dynamic	4096	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;
			}

Program we used to answer both empirical questions:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/sem.h>
_Bool shmPredicate(long sharedMemorySize);
_Bool semPredicate(long counting);
long shmLimit(_Bool (*predicate)(long));
int approximate(int a,int b);
key_t key; int getKey();
int semid;
void main(){
	getKey();
	if((semid=semget(IPC_PRIVATE,1, 00600))<0){
		perror("Failed to create semaphore");
	}

	printf("%lu¥n",shmLimit(shmPredicate));
```

```
	printf("%lu",shmLimit(semPredicate));
	if(semctl(semid,0,IPC_RMID,0)==-1){
			perror("Failed to remove");
		}

}
long shmLimit(_Bool (*predicate)(long)){
long j=0;
for(long i=0; i!=1; j=j+i/2){
	for(i=1;((*predicate)(i+j));i=i*2);
}
	return j;
}

_Bool semPredicate(long count){
	if((semctl(semid, 0, SETVAL,count)) == -1) {
	return 0;}

	return 1;
}
//This method needs some fixing
_Bool shmPredicate(long sharedMemorySize){
	int shmid;
	if((shmid=shmget(key,sharedMemorySize,IPC_CREAT | 0600))>0){
		shmctl(shmid,IPC_RMID,NULL);
		return 1;}
	return 0;
}

int getKey(){
	if(key=ftok("./",1)<1){
		perror("Failed to assign shmid");
		exit(1);
	}
}
```