Memory.h

Added struct memblk *gbgnext and pid32 gbgpid entries to struct memblk. Redefined roundmb and truncmb, changed 7 to 15 to account for the change in size of memblk. Added memblk gbglist to keep track of allocated memblk's.

Mysignal.h

Defined MYSIGRECV, MYSIGALRM, MYSIGXCPU, and declared registercbsig.

Process.h

Added alarmtime, alarmfunc, xcputime, and xcpufunc entries to procent

Xinu.h

Included mysendqueue.h and mysignal.h

Clkhandler.c

Added code to handle xcpufunc time.

Freemem.c

Added code to remove gbg entries from memblks that are being freed.

Getmem.c

Added code to keep track of allocated memory

Initialize.c

Declare gbglist, initialize new procent entries.

Kill.c

Added code to go through gbglist, freeing all memblks whose pid matches the pid passed in.

Meminit.c

Added code to initialize gbglist and memlists gbg entries

Registercbsig.c

Created registercbsig

Resched.c

Added code to handle alarmfunc.

Part1

4 new entries were added to procent to keep track of xcpu and alarm signals. Registercbsig is a simple modification of registercb, if MYSIGRECV is passed, then it simply acts as registercb, if MYSIGALRM or MYSIGXCPU is passed, it first checks to ensure optarg is not less than 0, and then sets the appropriate time and func entries of the processes procent.

For MYSIGRECV, it is the exact same as registercb. For MYSIGALRM, whenever the process is switched out in resched, clktimefine is checked and compared to the process alarmtime, if it is greater or equal to, then the alarm function is called. For MYSIGXCPU, xcputime is set to optarg minus cpuused time, and xcputime is then updated in clkhandler every millisecond, ensuring 1 millisecond granularity, and when xcputime is 0, xcpu func is called.

Part2

Memory Garbage Collection was implemented by adding 2 new entries to the memblk struct, and declaring a new memblk list called gbglist. Whenever memory is allocated (getmem is called), a pointer is set to the head of gbglist, and loops through until it reaches the end of list, where the new block of memory is then set to be the end of the list, the memblk gbgpid is set to the pid of the process that made that call, and the gbgnext is set to NULL, since it is the end of the list. Whenever a new memblk is allocated, gbgnext will be updated accordingly.

Whenever a block of memory is freed, prevgbg is set to the head of the list, nextgbg is set to the next entry, and then prevgbg and nextgbg are updated until either the end of the gbglist is reached, or gbgnext is equal to block of memory(block) that is being freed. If block is found, nextgbg is removed from gbglist, gbglist mlength is updated, block's gbgnext is set to NULL and gbgpid is set to -1.

Kill was also modified, so that the entire gbglist is looped through, and each entry in the gbglist whose pid matches the pid being killed has freemem called on them. Nextgbg is then set to the next entry in the list, and looping through continues. If the pid does not match, it continues searching through.