Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

ICS-OS Lab 03

---

## Environment Variables, Processes, and Threads

```
Your branch is up to date with 'origin/master'.
PS C:\Users\Gwy\Desktop\ics-os-mgat\ics-os> git pull
Already up to date.
PS C:\Users\Gwy\Desktop\ics-os-mgat\ics-os> git checkout -b lab03
Switched to a new branch 'lab03'
PS C:\Users\Gwy\Desktop\ics-os-mgat\ics-os> git branch
  lab01
  lab02
* lab03
  master
```

## Task 1: Environment Variables

**QUESTIONS:**

1. What data structure is used for the implementation of environment variables?

ics-os-mgat > ics-os > kernel > process > C environment.h

```c
1  /*
2    Name: environment.c
3    Copyright:
4    Author: Joseph Emmanuel DL Dayo
5    Date: 04/01/04 05:04
6    Description: Handles environment strings with concurrency support
7      As of the moment, implementation is on a global basis only, a per process
8      implementation will be implemented in the future.
9  */
10
11 #ifndef __ENVIRONMENT_H__
12 #define __ENVIRONMENT_H__
13
14
15 //A node in the environment variables data structure
16 //which is a doubly-linked list
17 typedef struct _env_strings {
18     char *name;                  //name of the variable
19     char *value;                 //value of the variable
20     struct _env_strings *next;   //pointer to the next node
21     struct _env_strings *prev;   //pointer to the previous node
22 } env_strings;
23
24 int env_busywait = 0;           //used for synchronization
25 env_strings *env_head = 0;      //head of the list
26
27 /* function prototypes*/
28 void env_showenv();
29 char *env_getenv(const char *name, char *buf);
30 int env_setenv(const char *name, const char *value, int replace);
31 #endif
```

The data structure used to implement the environment variables is a doubly-linked list. Lines 20 and 21 indicate that `_env_strings` contains pointers to its next and previous nodes.

Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

2. Are environment variables unique for each process or shared by all processes?

The environment variables are shared by all processes, this is also backed up by the author's documentation:

ics-os-mgat > ics-os > kernel > process > environment.h

```c
/*
  Name: environment.c
  Copyright:
  Author: Joseph Emmanuel DL Dayo
  Date: 04/01/04 05:04
  Description: Handles environment strings with concurrency support
    As of the moment, implementation is on a global basis only, a per process
    implementation will be implemented in the future.
*/
```

3. What are the functions used to set and get an environment variable?

ics-os-mgat > ics-os > kernel > process > environment.h

```c
27  /* function prototypes*/
28  void env_showenv();
29  char *env_getenv(const char *name, char *buf);
30  int env_setenv(const char *name, const char *value, int replace);
31  #endif
32
```

Lines 29 and 30 show the functions for the set and get environment variables, respectively.

4. Examine kernel/console/console.c. What console commands use the functions in question 3?

For setting and getting environment variables, the console commands "set" and "cc" are used.

ics-os-mgat > ics-os > kernel > console > console.c

```c
529  int console_execute(const char *str){
850     if (strcmp(u,"time") == 0){   //-- Displays
851        printf("%d/%d/%d %d:%d.%d (%d total seconds since 1970)\n",time_systime.day,
854                 time_systime.sec,time());
855     }else
856     if (strcmp(u,"set") == 0){    //-- Sets an environment variable. Args: <key>=<value>
857        u=strtok(0," ");
858        if (u==0){
859           env_showenv();
860        }else{
861           char *name  = strtok(u,"=");
862           char *value = strtok(0,"\n");
863           env_setenv(name, value, 1);
864        };
```

Setting an environment variable can be performed with the command "set".

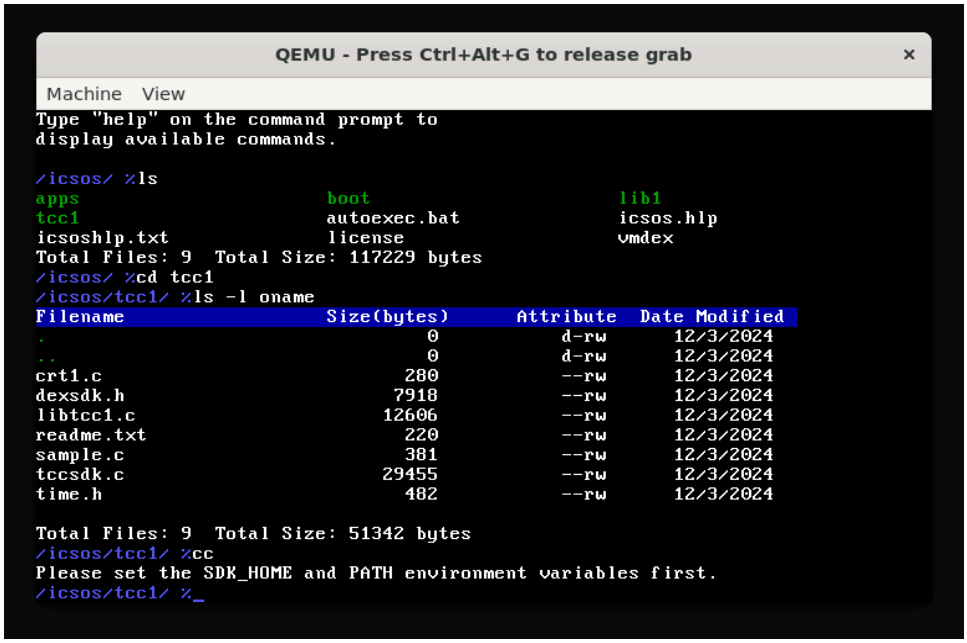
```c
876     if (strcmp(u,"cc") == 0){   //-- Builds a C program (invokes tcc.exe). Args: <name.exe> <name.c>
877        char src[30],exe[30],cmdline[256],path[256];
878        char sdk_home[128]="";
879        env_getenv("SDK_HOME",sdk_home);
880        env_getenv("PATH",path);
```

And writing a C program is done with "cc".

Mary Gwyneth A. Tolentino
2021-02990

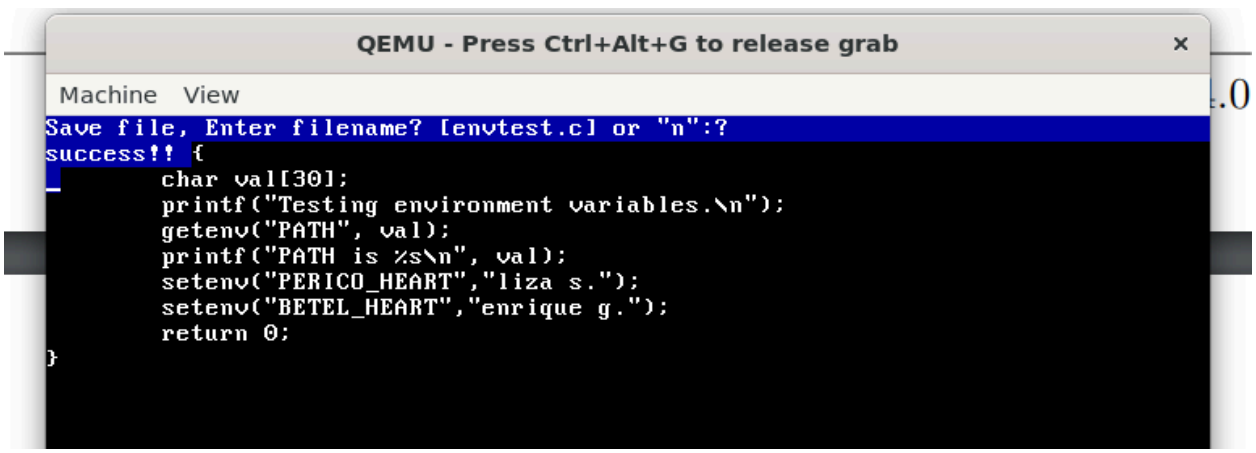
December 3, 2024
CMSC 125 - B5L

**Task 2: Editing and compiling programs within ICS-OS**




*Compiling a C program without setting SDK_HOME and PATH environment variable initially*

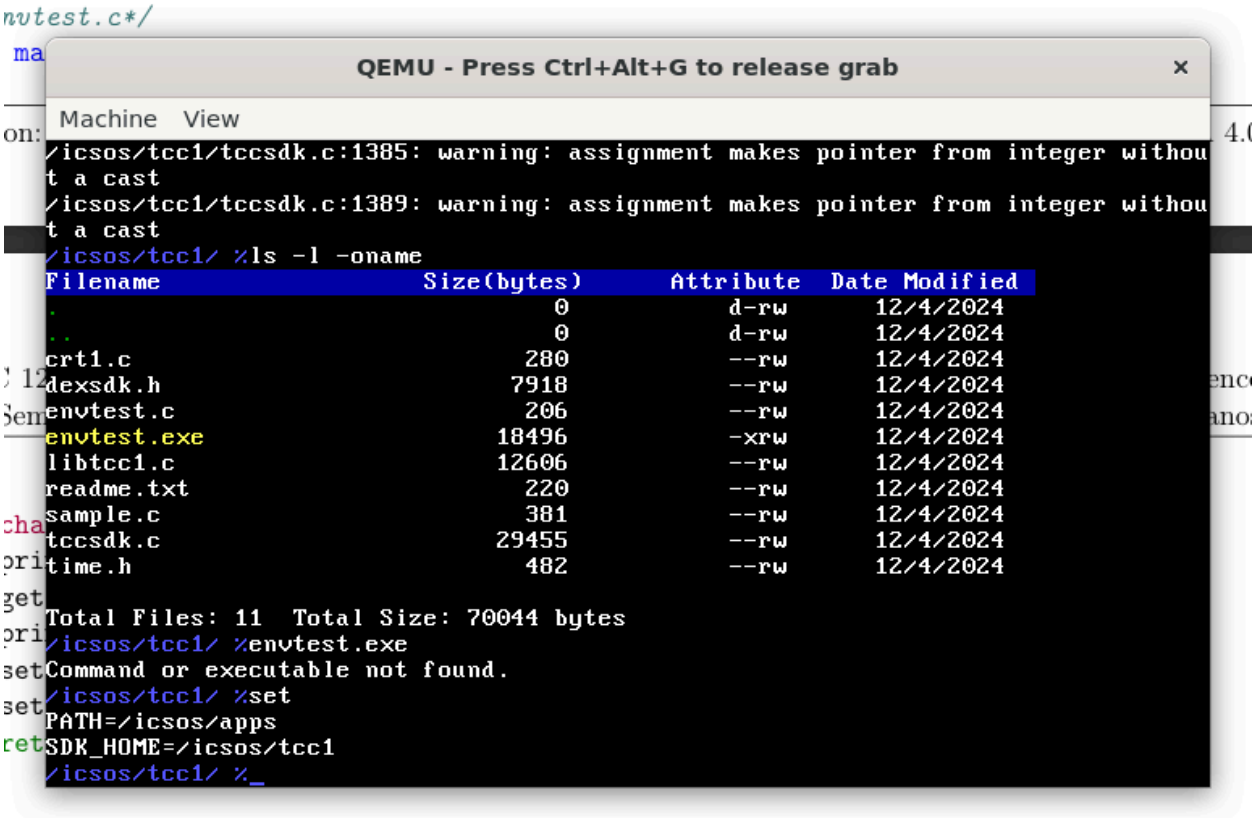

*Setting the value for the SDK_HOME and PATH environment variables*




*Writing envtest.c program using ed.exe (text editor for ICS-OS)*

Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L




*Attempting to run envtest.exe after compiling the C program*

**QUESTIONS:**

1. Observe that you were able to run ed.exe in command line [9] without specifying its absolute path despite the current directory being /icsos/tcc1. Why is this so?

Running ed.exe without specifying its absolute path is possible because executables can be accessed by default via the path provided in the environment variable "PATH", as shown in kernel/console.c. Otherwise, they will not be accessible through any directory.

ics-os-mgat > ics-os > kernel > console > C console.c

```c
int console_execute(const char *str){
    }else{          //ok it is not a command, maybe it's an executable?
        if (u!=0){
            if (strcmp(path,"")==0){
                strcpy(path,"/icsos/apps");
                sprintf(tmp,"%s/%s",path,u);
                if (!user_execp(tmp, 0, str)){
                    printf("Command or executable not found.\n");
                }
            }else{
```

```
/icsos/ %set
PATH=/icsos/apps
SDK_HOME=/icsos/tcc1
/icsos/ %_
```

Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

2. Command line [12] will not work. Why? Show your fix to be able to run envtest.exe.

It did not work because the executable was created within tcc1. To function, executables must be located in /icsos/apps, as specified by the set PATH=/icsos/apps command in previous instructions. I only compiled envtest.c in /icsos/apps to run envtest.exe successfully.

```
QEMU - Press Ctrl+Alt+G to release grab
Machine  View
Filename                Size(bytes)     Attribute  Date Modified
.                                 0          d-rw      12/4/2024
..                                0          d-rw      12/4/2024
chown.exe                     20216          -xrw      12/4/2024
ed.exe                        53194          -xrw      12/4/2024
envtest.c                       207          --rw      12/4/2024
envtest.exe                   18496          -xrw      12/4/2024
hello.exe                     20120          -xrw      12/4/2024
hxdmp.exe                     20236          -xrw      12/4/2024
lzozip.exe                    24024          -xrw      12/4/2024
nasm.exe                     308736          -xrw      12/4/2024
pak.exe                       17099          -xrw      12/4/2024
tcc.exe                      186056          -xrw      12/4/2024
vgademo.exe                   20472          -xrw      12/4/2024

Total Files: 13  Total Size: 688856 bytes
/icsos/apps/ %envtest.exe
Testing environment variables.
PATH is /icsos/apps
/icsos/apps/ %set
BETEL_HEART=enrique g.
PERICO_HEART=liza s.
PATH=/icsos/apps
SDK_HOME=/icsos/tcc1
/icsos/apps/ %
```

3. After successfully running envtest.exe. What is the output of command line [13]? Does this support your answer in Q2 from Task 1?

```
/icsos/apps/ %set
BETEL_HEART=enrique g.
PERICO_HEART=liza s.
PATH=/icsos/apps
SDK_HOME=/icsos/tcc1
/icsos/apps/ %
```

This supports my answer from Task 1, since environment variables are indeed shared globally, allowing multiple processes to access and set them.

## Task 3: Processes

### Task 3.1: Process Control Block

**QUESTIONS:**

1. What field in the PCB describes the security bits for a process?

Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

```
ics-os-mgat > ics-os > kernel > process > C process.h
161  typedef struct _PCB386 {
181
182      vfs_node *workdir;  //points to the vfs_node of the working directory
183
184      DWORD accesslevel;  //the security bits for this process (kernel/system or user)
185
```

2. What field in the PCB describes the time the process arrived in the system?

```
ics-os-mgat > ics-os > kernel > process > C process.h
161  typedef struct _PCB386 {
197      DWORD lastcputime, totalcputime; /*the taskswitcher increments this value every time
198                                          the process takes control of the CPU*/
199
200      DWORD arrivaltime;   //holds the time when this process arrived
201
```

3. What field in the PCB describes the memory information used by a process?

```
ics-os-mgat > ics-os > kernel > process > C process.h
161  typedef struct _PCB386 {
203
204      process_mem *meminfo; /*points to a data structure containing the memory locatons taken up by
205                              this process so that the process manager could clean this up easily.*/
206
```

4. What field in the PCB describes the execution context(hardware specific) of a process?

```
ics-os-mgat > ics-os > kernel > process > C process.h
155  /*The primary data structure used for describing a process or thread,
160  */
161  typedef struct _PCB386 {
162      /*regs must be at the beginning of the PCB since the
163       TSS directly points to the beginning of this structure*/
164      saveregs regs;      /* TSS data, also for placing initial values for EAX,EBX etc.
165                           A very hardware specific data structure for the Intel x86 family*/
166
```

**Task 3.2: Startup Processes**

**QUESTIONS:**

1. How many processes and kernel threads (those with (t) in the name) in total are running?

There are 5 processes in total, with 4 of them as kernel threads.

```
/icsos/ %ps
dex32_scheduler  v1.00
Processes in memory:

PID    Name              Access Lvl  PPID            Size     AT          CT
[0  ] dex_kernel         kernel      0                 4K     0s     53s(30)%
[16 ] task_mgr      (t)  kernel      0                 4K    23s     30s(17)%
[17 ] disk_mgr      (t)  kernel      0                 4K    23s     30s(17)%
[18 ] fg_mgr        (t)  kernel      0                 4K    24s     30s(17)%
[19 ] console(0)    (t)  kernel      0                 4K    24s     30s(17)%

Total              : 5 processes (20 KB)
Time Since Startup : 176
Legend: AT = Arrival Time, CT = CPU Time, %CT = Percent CPU Time
```

2. What is the name of the process with PID 0?

The name of the process with PID 0 is dex_kernel.

3. What is the PID of console(0)? What is its access level?

The PID is 19 and the access level is kernel level.

Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

4. What function is used to create the running kernel threads?

The function `dex_init()` is used to create the running kernel threads

```
ics-os-mgat > ics-os > kernel > C kernel32.c
void dex_init(){
    current_process->workdir= vfs_root;

    //Initialize the task manager - a module program that monitors processes
    //for the user's convenience, as kernel thread
    printf("Initializing the task manager...");
    tm_pid=createkthread((void*)dex32_tm_updateinfo,"task_mgr",3500);

    printf("[OK]\n");


    //create the IO manager thread which handles all I/O to and from
    //block devices like the hard disk, floppy, CD-ROM etc. see iosched.c
    printf("Initializing the disk manager...");
    createkthread((void*)iomgr_diskmgr,"disk_mgr",200000);
    printf("[OK]\n");
```

5. To what function is the EIP register assigned to in the PCB of the very first process?

The EIP register is assigned to `dex_init()` function from kernel32.c file.

```
ics-os-mgat > ics-os > kernel > process > C process.c
void process_init(){

    memset(&kernel->regs,0,sizeof(saveregs));          //initialize the execution context
    kernel->regs.EIP=(DWORD)dex_init;                  //dex_init() in kernel32.c
    kernel->regs.ESP= DISPATCHER_STACK_LOC;            //set the values of the registers for kernel
    kernel->regs.CR3=pagedir1;
    kernel->regs.ES=SYS_DATA_SEL;
```

**Task 3.3: Consoles**

**QUESTIONS:**

1. Using ps, what is the name and PID of the new console?What is the name and PID of its parent process? Is the new console a process or a thread?

```
/icsos/ %newconsole
New console thread created.
/icsos/ %ps
dex32_scheduler  v1.00
Processes in memory:

PID    Name              Access Lvl  PPID          Size     AT        CT
[0  ] dex_kernel         kernel      0              4K     0s    58s(19)%
[16 ] task_mgr      (t)  kernel      0              4K    23s    34s(19)%
[17 ] disk_mgr      (t)  kernel      0              4K    23s    34s(19)%
[18 ] fg_mgr        (t)  kernel      0              4K    24s    34s(19)%
[19 ] console(0)    (t)  kernel      0              4K    24s    34s(19)%
[20 ] console(1)    (t)  kernel      19             4K   194s     1s( 4)%

Total                : 6 processes (24 KB)
Time Since Startup : 200
Legend: AT = Arrival Time, CT = CPU Time, %CT = Percent CPU Time
```

The new console's name is console(1), and its PID is 20. Its parent process is console(0), with PID 19. Also, the new console is a kernel thread (as indicated by the (t) mark).

Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

2. Study the implementation of the newconsole command in the kernel/console/console.c. What function is used to create a new console?

```
ics-os-mgat > ics-os > kernel > console > C console.c
int console_new(){
    //create a new console
    char consolename[255];
    sprintf(consolename,"console(%d)", console_first);
    return createkthread((void*)console, consolename, 200000);
};
```

console_new  Aa ab .* 2 of 2

```
if (strcmp(u,"newconsole") == 0){   //-- Creates a new console.
    //create a new console
    console_new();
    printf("New console thread created.\n");
```

The command "newconsole" uses the function console_new() to call createkthread() with a console as its argument.

**Task 3.4: User Processes**

**QUESTIONS:**

1. What is the PID of count.exe process? What is its access level? How much memory does it use? What is its parent process?

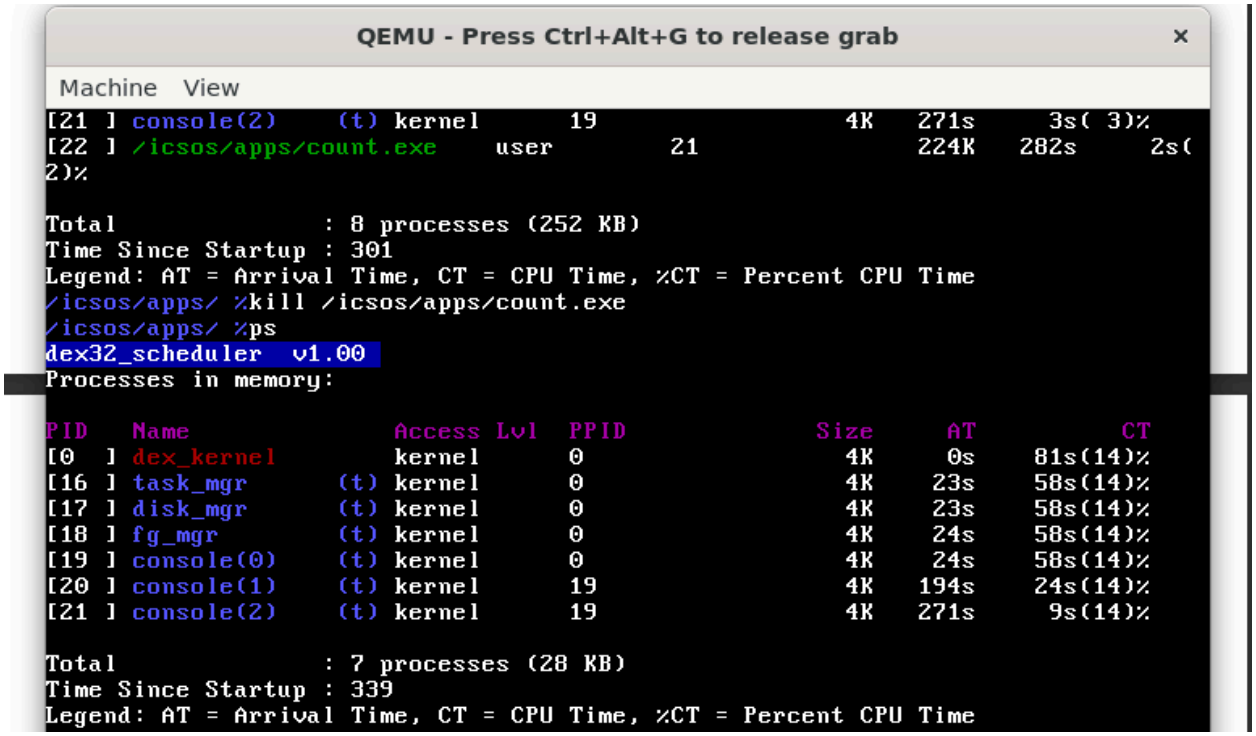
folder. Do not forget to set the needed environment variables described above.

QEMU - Press Ctrl+Alt+G to release grab

Machine View

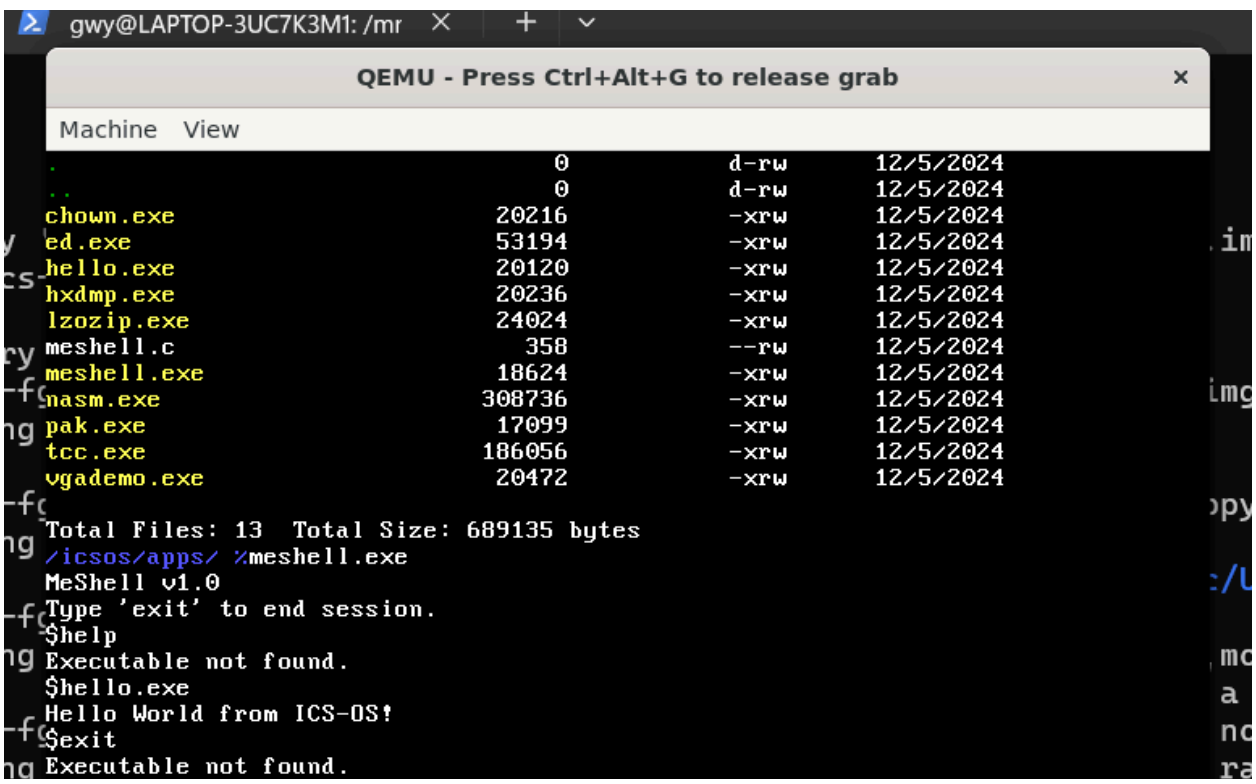
```
vmdex                          93090          --rw      12/4/2024

Total Files: 9  Total Size: 117229 bytes
/icsos/ %cd apps
/icsos/apps/ %newconsole
New console thread created.
/icsos/apps/ %ps
dex32_scheduler  v1.00
Processes in memory:

PID   Name              Access Lvl  PPID            Size     AT         CT
[0  ] dex_kernel        kernel      0                 4K     0s    76s(15)%
[16 ] task_mgr      (t) kernel      0                 4K    23s    52s(15)%
[17 ] disk_mgr      (t) kernel      0                 4K    23s    52s(15)%
[18 ] fg_mgr        (t) kernel      0                 4K    24s    52s(15)%
[19 ] console(0)    (t) kernel      0                 4K    24s    52s(15)%
[20 ] console(1)    (t) kernel      19                4K   194s    19s(15)%
[21 ] console(2)    (t) kernel      19                4K   271s     3s( 3)%
[22 ] /icsos/apps/count.exe    user        21              224K   282s     2s(
2)%

Total                : 8 processes (252 KB)
Time Since Startup : 301
Legend: AT = Arrival Time, CT = CPU Time, %CT = Percent CPU Time
/icsos/apps/ %
```

The PID of count.exe process is 22 with a user access level. It uses 224KB memory and its parent process is console(2) with PID 21, which was the recently created console used to run count.exe.




*Use kill console command to terminate the count.exe process*

**Task 3.5: Process Creation**



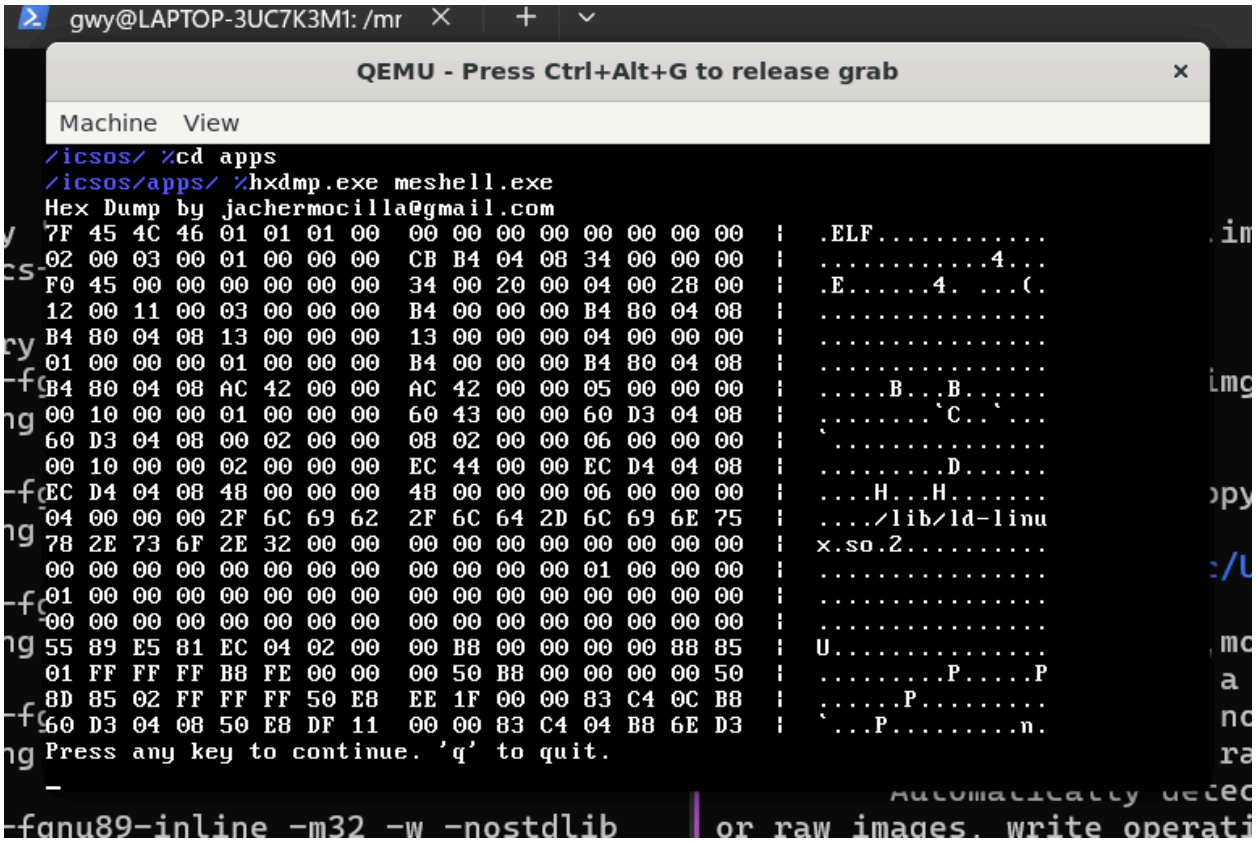

*Running meshell.exe inside ICS-OS*

Mary Gwyneth A. Tolentino
2021-02990

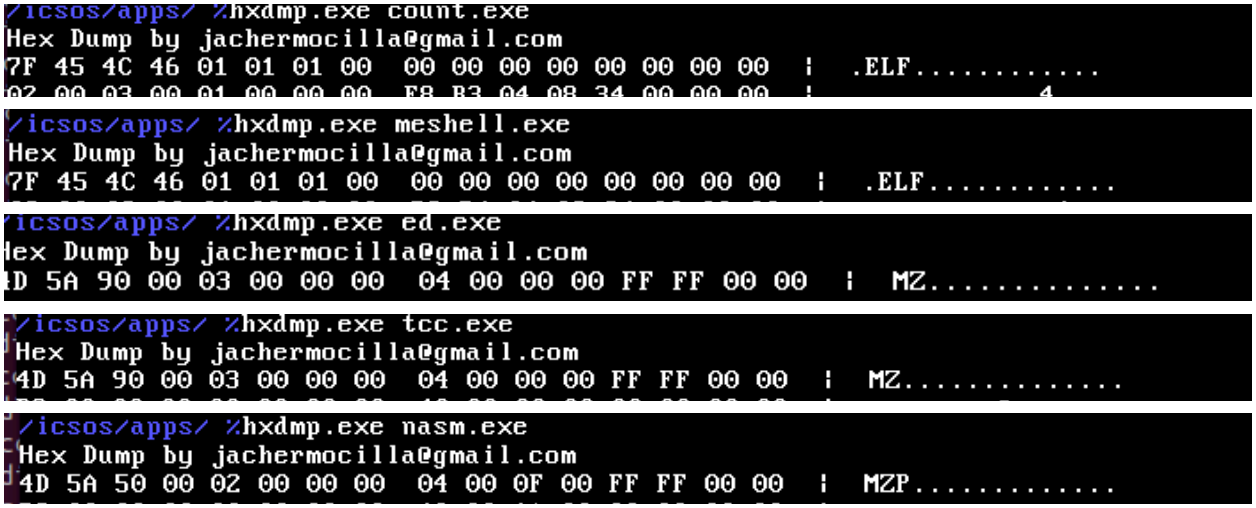
December 3, 2024
CMSC 125 - B5L

**QUESTIONS:**

1. Use hxdmp.exe to determine the format of some of the executables in the apps folder. If the first few bytes has MZ then it is a windows executable, if ELF then it is a linux executable. What is the executable format of count.exe? ed.exe? tcc.exe? nasm.exe? meshell.exe?




Upon examining each executable file according to the format shown in the screenshot (hxdmp.exe <executable file>), it is determined that count.exe and meshell.exe are Linux executables, whereas ed.exe, tcc.exe, and nasm.exe are Windows executables, as indicated by their first few bytes.



Mary Gwyneth A. Tolentino
2021-02990

December 3, 2024
CMSC 125 - B5L

2. Which line in the forkprocess() and createprocess() functions initializes the PCB of the new process. How do the functions differ?

The PCB is initialized within the createprocess at memset(temp, 0, sizeof(PCB386)), followed by memcpy(pcb, parent, sizeof(PCB386)).

ics-os-mgat > ics-os > kernel > process > C process.c

initialize 3 of 11

```c
DWORD forkprocess(PCB386 *parent){

#ifdef DEBUG_FORK
    printf("fork process has been called.\n");
#endif

    pcb = (PCB386*) malloc(sizeof(PCB386));//Allocate space for PCB
    dex32_stopints(&flags);                //disable interrupts
    memcpy(pcb,parent,sizeof(PCB386));     //Initialize the new process by copying the parent process'
    strcat(pcb->name,".fork");             //Add a 'fork' suffix to indicate that it was created by for
```

ics-os-mgat > ics-os > kernel > process > C process.c

initialize 5 of 11

```c
DWORD createprocess(

    PCB386 *temp=(PCB386*)malloc(sizeof(PCB386));      //allocate the PCB for the process
    memset(temp,0,sizeof(PCB386));                     //Initialize by zeroing it out
    temp->before=current_process;                      //add it after the current process
    strcpy(temp->name,name);                           //set the name of the process
    totalprocesses++;                                  //increase the total number of processes in the
```

Essentially, I think the main difference is that `forkprocess()` copies its parent process to initialize a PCB, whereas createprocess does not do so.

```
/icsos/apps/ %ps
dex32_scheduler  v1.00
Processes in memory:

PID   Name                Access Lvl  PPID            Size     AT          CT
[0  ] dex_kernel          kernel      0                 4K     0s    127s(11)%
[16 ] task_mgr       (t)  kernel      0                 4K    23s    104s(11)%
[17 ] disk_mgr       (t)  kernel      0                 4K    23s    104s(11)%
[18 ] fg_mgr         (t)  kernel      0                 4K    24s    103s(11)%
[19 ] console(0)     (t)  kernel      0                 4K    24s    103s(11)%
[29 ] console(1)     (t)  kernel      19                4K   596s      9s(11)%
[30 ] /icsos/apps/meshell.exe     user        29              224K   607s      8s
(11)%
[31 ] fork.exe            user        30              224K   613s      7s(11)%
[32 ] fork.exe.fork       user        31              444K   613s      7s(11)%

Total                : 9 processes (916 KB)
Time Since Startup : 680
Legend: AT = Arrival Time, CT = CPU Time, %CT = Percent CPU Time
```

*Running ps after fork.exe*

**Task 3.6: Process Termination**

**QUESTIONS:**

**Study the kill_process() function.**

1. What function is called to kill a kernel process/thread?

The function called to kill a kernel process/thread is `dex32_killkthread(ptr)`

```c
DWORD kill_process(DWORD processid){
    PCB386 *ptr,*parentptr=0;

    sync_entercrit(&processmgr_busy);                       //Try to enter the critical section

    ptr = bridges_ps_findprocess(processid);                //obtain a handle to the PCB given id

    if (ptr!=-1){                                           //PCB exists

        if (! (ptr->status & PS_ATTB_UNLOADABLE) ){

            PCB386 *parent;

            kill_children(processid);                       //kill children processes first

            if (ptr->accesslevel == ACCESS_SYS){            //a kernel process/thread
                dex32_killkthread(ptr);
```

2. What function is called to kill a user thread?

The function called to kill a user thread is `kill_thread(ptr)`

ics-os-mgat > ics-os > kernel > process > C process.c

kill Aa ab .* 10 of 29

```c
//kill user threads
DWORD kill_thread(PCB386 *ptr){
    DWORD flags;
    dex32_stopints(&flags);

    kill_children(ptr->processid); //kill the children of this thread first!!cascade kill

    //Tell the scheduler to remove this thread from the ready queue
    ps_dequeue(ptr);

    if (ptr->stackptr0!=0)
        free(ptr->stackptr0);

    free(ptr);
    dex32_restoreints(flags);
    return 1;
    ;
};
```

---

**REFLECTION**

This lab activity demonstrated the environment variables and threads in ICS-OS. Despite my previous knowledge from lectures and labs, I just learned that environment variables can be used to access files outside of the working directory. I think this is beneficial because it allows users to easily access frequently used files from any location. Environment variables store directories, making them a useful shortcut that can be shared globally. Furthermore, I discovered that the ICS OS does not yet support the limitation of variables to specific processes. Using the ps command, I gained insight into the activity of my operating system by monitoring active processes and threads. This lab activity gave me a better understanding of how processes and threads work in the CLI. I also noticed that the authors completed this project in 2003, 2004, or 2005, and while it was already out of date, I believe the topics covered in this project are fundamental to operating systems and are still used in modern operating systems.