

1) Write a program that takes one command line argument, long int n. The main() function calls a recursive function int recurse(long int n). In the body of the recursive function, use malloc to dynamically allocate space for storing 10 characters and store "CDPCS-310" to that allocated memory. Call the recursive function with n-1. The base condition for the recursive function is n=0.

A) The program should output:

- 1) address of the main function and the recursive function
e.g. printf("_main @ %lx\n", (long unsigned int)&main);
- 2) address of allocated memory in the heap at each recursive call
- 3) address of the parameter passed to the recursive function

Based on the above answer:

- 1) Which of the addresses printed by your program are defined in the output of objdump "objdump -sRrd a.out"?
- 2) What direction is the stack growing in?
- 3) How large is the stack frame for each recursive call?
- 4) Where is the heap? What direction is it growing in?
- 5) Are the two malloc()'ed memory areas contiguous? (e.g. is there any extra space between their addresses?)

B) Load up your program executable in gdb, set a breakpoint at main. Continue 1 line at a time. When you have finished n/2 recursive calls, take a look at the stack using backtrace (bt).

- 1) Type "info address main"
- 2) Type "info stack" in gdb. Explain what you see.
- 3) Type "info frame". Explain what you see.

2) #include <stdio.h>
#include <unistd.h>

```

int main()
{ int i;
  for(i=0; i<5; i++)
    if (fork())
      fork();
  return 0;
}

```

How many processes are created by this program? Add appropriate print statements to find out the number of processes.

3)

```

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

```

```

int main(){
  int id ;
  int counter = 100;
  id = fork() ;
  if ( id == 0 ) {
    printf ("\nCHILD::PID: %d", getpid());
    counter = counter+200;
  }else {
    printf ( "\nPARENT::PID: %d", getpid());
    counter = counter +50;
  }
  printf ("\ncounter: %d, PID: %d", counter, getpid());
  return 0;
}

```

What is the value of counter? From the value, what do you conclude: is counter a shared variable between the parent and the child? hint: process gives memory protection.

4) Write a program that takes an integer K (> 1000000) as input and does the following:

- a) Creates 2 child processes, and 4 grand children processes.
- b) The child processes compute the prime numbers between $[2, K/2)$ and $[K/2, K)$ respectively, print "Hello from <pid>, I have computed prime numbers from <start , end >, my parent pid is <ppid>" and then exits.
- c) The grand children processes wake up after every sec and print "Hello from <pid>, I have no work. My parent pid is <ppid>" They do this 10 times and exit.
- d) i) The original process waits until all of them exit.
ii) The original process doesn't wait until all of them exit.

Use `ps tree` command with the pid of the original process and see the hierarchy of processes.

Questions:

What difference do you observe in d(i) and d(ii)? Look at the parent id in the output. Is it same as the id of the parent which created it?