Lab Work Week 3

Create a ls program using C

getcwd()
These functions return a null-terminated string containing an absolute pathname that is the current working directory of the calling process. The pathname is returned as the function result and via the argument buf, if present.

The getcwd() function copies an absolute pathname of the current working directory to the array pointed to by buf, which is of length size.

```
int scandir(const char *dirp,
        struct dirent ***namelist,
        int (*filter)(const struct dirent *),
        int (*compar)(const struct dirent **, const struct dirent **)
);
```

The scandir() function scans the directory dirp, calling filter() on each directory entry. Entries for which filter() returns nonzero are stored in strings allocated via malloc(3), sorted using qsort(3) with the comparison function compar(), and collected in array namelist which is allocated via malloc(3). If filter is NULL, all entries are selected.

// Code example below

To do: can you find a solution to the warning for getwd warning??

```
#include <sys/types.h>
#include <sys/dir.h>
#include <sys/param.h>
#include <stdio.h>
#include <stdlib.h>

// List of functions
int file_select();
extern int alphasort();

main()   {
  int count,i;
  struct direct **files;

  char pathname[MAXPATHLEN];
  int t = 1;

  getwd(pathname);

  printf("PWD = %s\n",pathname);
  count = scandir(pathname, &files, file_select, alphasort);

  /* No files in Dir */
  if (count <= 0)
  {
    printf("No files in Dir\n");
    exit(0);
  }

  printf("Number of files = %d\n",count);

  for (i=1;i<count+1;++i)
  {
   printf("\n%s",files[i-1]->d_name);
  }

  printf("\n");
}

int file_select(struct direct *entry)
{
    return (1);
}
```

Create separate programs to offer the following functionality:

1. Write a C program to emulate the `ls -l` UNIX command that prints all files in a current directory. Hint: Use the exec command.

2. Write a C program to offer the following functionality:
   - **a.** List all the processes running on a system if no params are passed via the command line
   - **b.** Search for a process by name eg. ./myprog search calculator
   - **c.** Kill a process for a given IPD rg. ./myprog kill 1292

3. Create a program that manages the process of automatically creating a log file in a directory. This should be a separate process. If this process terminates unexpectedly the main program should be informed via an appropriate signal.

4. Create a program to ask a user for their name and then append this to a log file (text file).

5. Create a program to display all the names stored in the log file from part 4 above.

6. Create a program to change the permissions of the log file to 777.

7. Write a C program to emulate the `ls -l` UNIX command that prints all files in a current directory and lists access privileges etc. DO NOT use the `exec ls -l` from the program.

8. Expand the functionality of the program from step 7 to only list specific types. Eg, .c .doc .txt etc. The program also should not display the . and the .. when listing the file names.