Hacettepe University
Computer Science and Engineering Department

Name and Surname	**:** Ozan Özenoğlu
Identity Number	**:** 20724945
Course	**:** Bil-235
Experiment	**:** Experiment 3
Subject	**:** C Programming in Unix
Data Due	**:** 25.11.2010
Advisors	**:** Dr. Ahmet Burak CAN , R.A Safa SOFUOĞLU
e-mail	**:** b20724945@cs.hacettepe.edu.tr
Main Program	**:** ichk

Software Using Documentation:

To start software firstly you need to start terminal in unix operating system. After this operation you need to go directory that include program make file. After that you need to enter “make” command into console to generate executable file.After make command the executable ichk file will generate by make utiliy. Now you can start program with this command :

./ichk /path delayNumber İterationTime. DelayNumber and İterationTime are parameters for program. You need to type them as a integer .

Provided Possibilities: There is no extra utilities except main goal of this program.

Error Messages : There is no error message that can given by program so we assume user will enter in perfect correct parameter in commond line. More about parameters info please read below (Section System Chart)

Software Design Notes: This scope is for software developers who interest the developing monitoring program in unix operating systems.

Description of the program

Problem : We need to monitor all operation with all files that belong given directory with all subdirectories. And then report any operation into log file. User may want to run program recursively every $n seconds for a total $n iterations. So we need to handle this station in program.

Solution : In start ı call monitoring function immediately to save starting info of all files into ichk.sum .To produce recursively running program with a delay , I use time function in time.h library. When start program in main function firsly ı save current time as a start time. After create a variable as difference and in a loop program calculates difference between start time and current time into difference variable. When difference equals delay ı call monitoring function in loop and decrease one from iterations count.

In Monitoring function , ı used some structure and its List. This function take a file path as a parameter. After that call direc_reader function with this paremeter. Direc Reader function parse all files name and return char* structure back. After this operation program call stat function which include in stat.h library . So ı can learn inode number and file size with this function.After that program opens file and read the data from file. Then calls adler32 function to calculate checksum info of data. After all these data collection operation programs saves these info in infoList. And MyList. MyList and infoList are similar except one difference. MyList can handle file name. For more information about structure and their list please read below (Section 3.3 Main Data Structures). While all these operation is doing if program catch a directory besides file . Calculate its pathname and call itself. So this function runs recursively for subdirectories. This function also reports any operation on files . This function take a file info from infoList and call find file function. İf function can find file , match checksum info. İf does not match reports this file changed. İf function can't file , reports a new file added. After this operation the monitoring function reads old info from current sum. And match old info with current info. İf there is no match function reports deleted.

System Chart

INPUT

input device: keyboard , 2 nd memory
input file : ichk.sum
other inputs : arguments

arguments formation : %s %d %d
an example argument : /workspace/ 5 10
the first integer is delay
the second integer is iteration count.

Ichk.sum is a file that does'nt consider user.
We store files information in this file.
Program read and write into this file.
All directories* includes this file.

OUTPUT
output device: monitor
output file: ichk.log
other outputs: There is no other outputs.
output formation %s %s
an example output: file_name (added || changed) or inode deleted
the first string is file_info the second one is operation info
Ichk.log is a file for users.
All operation information for each directory saves into ichk.log
All directories* includes this file.

***All directories:** The path* and its subdirectories
***The path :** The file path that given as a argument by user in command line.

Main Data Structures:

These are code of structures that ı used in C language.

```
typedef struct {
	ino_t inode_number ;
	unsigned long checksum ;

}ichk_info;

typedef struct {
	ino_t inode_number;
	char file_name[32];
}my_file_struct;
```

ichk_info : This struct has two variable. First variable's type is ino_t and it's name is inode_number; Second variable's type is unsigned long and it's name is checksum.

In unix system every file have a inode number. And we save this inode number in inode_number. Every file may have a data and we calculate a checksum information from this data by adler32 algorithm. Any changes in data will change this checksum information. İf any data is not same a data their checksum information can't be same.

my_file_struct : This struct has two variable. First variable's type is ino_t and it's name is inode_number; Second variable's type is char* it's name is file_name.

We already know what is inode number now. I will explain variable that type is char* . I use this struct to store which file has which inode number. So when a file added ı use this struct to get file name.

Algorithm :

Main function:
Take arguments
get start time
calculate difference between current and start time.
In every difference %delay == 0 call monitoring function.

Monitor function Algorithm:
Get parsed file name by calling direc_reader function:
save inode number and checksum information into infoList(struct ichk_info)
read old informatiom from current sum file.
Match new information and old information in each other.
Report any operation into log file.

Special Design Properties :

İn this Experiment I use unix system call to open , close , read and write operation .
I take care of modular programming also.

Execution Flow Between Subprograms

```
ichk_info findFile(ino_t inode, int sum);

this function search ickh_info that belong specified inode number
return ichk_info of file.
Called in monitoring function
calls lseek and read function.

int open_file(char * filePath , char** fileList , int totalFile,char * fileName ,
int * isNew);

this function search file in specified by name in directory specified by filepath.
Return file number
called in monitoring function
calls strcpy, strcmp , strcat and open function.


unsigned long adler32(unsigned char *data, int len);
this function calculate checksum of file.
Returns checksum information of file
called in monitoring function
calls nothing
```

```
char ** direc_reader(char* directoryPath , int * i)
```

This function parse all filename in the directory that specified by directoryPath.
Returns a list of parsed named as char*
called in monitoring function
calls readdir, opendir, strcmp ,

```
void monitor(char* filePath );
```

This function monitor directory with its subdirectories and report any operation in these directories.
Returns nothing
called in main function and inself.
Calls direc_reader , adler32, open_file , find_file and other system call like open , write, close etc.

Software Testing Notes

Bugs and Software Reliability:
There is no bugs that ı can find.

Software Extendibility and Upgradability:
In future additional features may be added easily.

Performance Considerations
Program has a memory leak problem. For 120 iteration program take 1.6mb from memory. If ı have time to fix this ı will use free function to solve this problem.

Comments
This experiment is a window that opening linux experience. I love linux system call and also linux o.s. Everything is installed in linux. Linux is very safe and fast operating system. And also it's completely free.

REFERENCES : http://www.belgeler.org/howto/makefile-nasil-kullanimi.html
http://www.di.uevora.pt/~lmr/syscalls.html