```
/*-------------------------------------------------------------------------------------------------------------------
--	SOURCE FILE:	InotifyDaemon.c -	An application that will monitor a specified
--					directory for file creation/modification.
--
--	PROGRAM:	inotd
--
--	FUNCTIONS:
--			void daemonize (void)
--			int initialize_inotify_watch (int fd, char pathname[MAXPATHLEN])
--			int ProcessFiles (char pathname[MAXPATHLEN])
--			unsigned int GetProcessID (char *process)
--
--
--	DATE:		March 16, 2008
--
--	REVISIONS:	(Date and Description)
--
--	DESIGNER:	Aman Abdulla
--
--	PROGRAMMER:	Aman Abdulla
--
--	NOTES:
--	The program will monitor a directory that is specified in a configuration file for any type of file
--	modification activity (creation, read/write, deletion).  The design uses the “inotify” kernel-level
--	utility to obtain the file system event notification.  The “select” system call is used to monitor
--	the watch descriptor (returned from inotify).
--
--	Once select is triggered, the directory under watch is processed to determine the exact type of
--	file activity. Once the created/modified files have been identified, they are moved to a separate
--	archive directory. Before the archival process takes place, the system process table (/proc) is
--	searched to verify that the modifying process is currently active and running.
--
--	Note that the application once invoked, will continue to execute as a daemon.
---------------------------------------------------------------------------------------------------------------------*/
```

```
/*------------------------------------------------------------------------------------------------------------------
--	FUNCTION:	daemonize
--
--	DATE:		March 16, 2008
--
--	REVISIONS:	(Date and Description)
--
--	DESIGNER:	Aman Abdulla
--
--	PROGRAMMER:	Aman Abdulla
--
--	INTERFACE:	void daemonize (void)
--
--	RETURNS:	void.
--
--	NOTES:
--	Call this function to “daemonize” the application. Basically this function forks a new process,
--	allows the parent process to exit after setting the umask on the child process as global.
------------------------------------------------------------------------------------------------------------------*/


/*------------------------------------------------------------------------------------------------------------------
--	FUNCTION:	initialize_inotify_watch
--
--	DATE:		March 16, 2008
--
--	REVISIONS:	(Date and Description)
--
--	DESIGNER:	Aman Abdulla
--
--	PROGRAMMER:	Aman Abdulla
--
--	INTERFACE:	int initialize_inotify_watch (int fd, char pathname[MAXPATHLEN])
--				int fd: the descriptor returned by inotify_init()
--				char pathname[MAXPATHLEN]: fully qualified pathname of
--					directory to be watched.
--
--	RETURNS:	Returns the watch descriptor (wd), which is bound to fd and the
--			directory pathname.
--
--	NOTES:
--	This function is used to generate a watch descriptor using a initialized descriptor from
--	inotify_init and a specified pathname. This watch descriptor can then be used by the select call
--	to monitor for events, i.e., file activity inside the watched directory.
------------------------------------------------------------------------------------------------------------------*/
```

```
/*-----------------------------------------------------------------------------------------------------------------
--      FUNCTION:   ProcessFiles
--
--      DATE:               March 16, 2008
--
--      REVISIONS:          (Date and Description)
--
--      DESIGNER:           Aman Abdulla
--
--      PROGRAMMER:         Aman Abdulla
--
--      INTERFACE:          int ProcessFiles (char pathname[MAXPATHLEN])
--                                  char pathname[MAXPATHLEN]: fully qualified pathname of
--                                          directory to be watched.
--
--      RETURNS:            Returns a count of the number of files modified in the directory.
--
--
--      NOTES:
--      This function is used to determine the number of files that were created and/or modified in the
--      directory under watch. The function uses the "scandir" system call to determine how many files
--      are active. It then makes sure that process that modified the files is active, and then archives the
--      active files to a new directory.
-------------------------------------------------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------------------------------------------------
--      FUNCTION:   GetProcessID
--
--      DATE:               March 16, 2008
--
--      REVISIONS:          (Date and Description)
--
--      DESIGNER:           Aman Abdulla
--
--      PROGRAMMER:         Aman Abdulla
--
--      INTERFACE:          unsigned int GetProcessID (char *process)
--                                  char *process: the name of the process to be validated.
--
--      RETURNS:            Returns the PID of process specified if the process exists.
--                          Returns 0 if the process was not found in the process table.
--      NOTES:
--      This function is used to determine the PID of a process, given the name of the process.
--      The function reads the process table in "/proc" to determine the PID of specified process name
--      and returns the PID to the calling process.
-------------------------------------------------------------------------------------------------------------------*/
```