CS 367 - Recitation #10

Name	G#
Group Member Name:	
Group Member Name:	

Today's Goals: We want to get comfortable with basic signal handling and Unix-I/O.

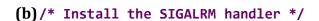
Work in groups of 2-3 students. Every group will turn in what they've got to Blackboard.

Grading is based on participation. Get as much done as you can. You will also be given feedback in the form of a 'score' (1-3) and possibly some comments. This doesn't affect your grade – it is solely for feedback. A score of 3 means everything looks great. A score of two indicates some minor problems. And a score of one indicates that there were some major issues. If you get a 1, don't panic - go see your prof or a GTA to get more extensive feedback.

Signal Handling (Chapter 8.5)

- 1. Explain below in plain words.
 - (i) What does it mean to "install" a signal handler?
 - (ii) What is "catching" or "handling" the signal?
- 2. Below is a simple alarm code using signal. Write the code corresponding to the boxes.

Reference: int sigaction(int sig, const struct sigaction *restrict act, struct sigaction *restrict oact); // Registers handler Note: struct sigaction needs to be initialized to 0s (hint: memset(void *s, int c, size_t n)) Note: struct sigaction has one important member: sa_handler to set to the handler function. void sighandler (int); // Signature for the handler (a) /*Ignore SIGINT (ctrl-c)*/ (hint: use SIG_IGN as the function to ignore a signal)



Unix I/O (Redirects) Chapter 10.9

- 3. Explain below in plain words.
 - (i) What does it mean to "redirect" a stream from stdout to a file?
 - (ii) What does the **dup2** function do?
- 4. Below is a simple redirect out sample program.

int open(const char *pathname, int flags,int mode); // Open a device and returns an FD Note: Flags...

- O_RDONLY read only
- O_WRONLY write only
- O_RDWR read and write
- O_CREAT If it doesn't exist, create a new file
- O_TRUNC If it does exist, erase it out before writing to it
- O_APPEND If it does exist, add the new stuff to the end

 $int\ execl (const\ char\ *path,\ const\ char\ *arg,\ ...);\ //\ path\ includes\ name,\ arg 0\ is\ the\ name,\ NULL\ to\ end\ the\ list.$

- The ... here means you can keep adding args one at a time, separated by a comma, until you're done.
- Then you use NULL to end it.
- Example: execl("/usr/bin/ls", "ls", "-a", "-l", NULL);
- execv is another version of exec that may be easier to use when actually programming. int dup2(int oldfd, int newfd); // Create a copy of the file descriptor.
 - Remember, oldfd is the one you want to install. newfd is where you want to install it.
 - Eg. dup2(fd, STDIN_FILENO); // This will set STDIN of the process to fd.
 - Valid newfds are: STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO

