Unix Summary

- Process

#include <sys/types.h>
#include<unistd.h>
int pid=fork(): returns -1 if failed, 0 if succeeded or returns the PID of the child process to the parent process.

#include<sys/wait.h>
int wait(int * status) function suspends execution of its calling process until *status* information is available for a terminated child process, or a signal is received.

int execl(const char *path,const char *arg(), const char *arg1,…,const char *argn,NULL);
int execlp(const char *file,const char *arg(), const char *arg1,…,const char *argn,NULL);
int execv(const char *path,char *const argv[]);
int execvp(const char *file,char *const argv[]);
int execle(const char *path,const char *arg,…,char *const envp[]);

- Signal

#include <signal.h>
void *signal(int,func()); func אז מפעילים את הפונקציה sig_num אם מקבלים סיגנל
int sigblock(int mask); : עוצר את הסיגנל. אינו משפעי על הגדרות פונקצית התגובה. מחזיר 1 במקרה והצליח, אחרת 0.
int sigmask(signum int);: מחזיר 1 אם אותו סיגנל מוגדר כMASK כלומר הוא בלתי נגיש. אם 0 – לא בודד את הסיגנל המסוים.
sigblock(0);: מבטל את העצירות השונות שהגדרנו.
signal(מצביע לפונקציה שאומרת מה לעשות עם הסיגנל, זהות של אותו הסיגנל)
kill int(sig int ,pid pidt): שולח לתהליך מסוים את הסיגנל שרוצים לבצע.
raise int(sig int): כמו kill אבל שליחת הסיגנל לתהליך שרץ כרגע.

#include>stdlib.h<
void abort(void): ביטול. int pause(void): השהייה - unsigned int alarm(unsigned int seconds): (2)מחזיר כל פעם כמה זמן נשאר לתהליך לעבוד. לאחר זמן שקיבל כארגומנט בד"כ הורג את התהליך שרץ עכשיו.
alarm(0) - מנגנון שמבטל את ה alarm--
בנושא הסיגנלים – חשוב להכריז על הפונקציה שאומרת מה לבצע בסיגנל, עוד לפני הmain.

- Inter Process Communication (IPC(

int pipe(int *fd); //int pipe(int fd[2]);
אם מצליחים ליצור pipe, מחזיר מס' חיובי.אחרת מחזיר fd -1. לקריאה הוא בד"כ p[0] לכתיבה .p[1] – כללים בשימוש:pipe
1. read מבצע עצירה (לא מאפשר לקרוא בpipe). SIGPIPE קורה כאשר יש התנגשות בקריאה/כתיבה (לרוב).
2. גודל הזיכרון המוקצה לpipe הוא לרוב עד kb9, ניתן לשנות את גודלו.
3. pfile mknod

pipe broken: במקרה שאבא סיים עבודה שלו והבן עדיין כותב, הבן מנסה לכתוב אך האבא כבר סיים וסגר את הfd של הקריאה שלו. כלומר אם האבא מת לפני הילד יהיה broken pipe ויישלח סיגנל SIGPIPE

int dup)int oldd;(
int dup2(int oldd, int newd);
תפקידם לשנות הגדרה של fd קיימים ומגדירם לפעולות שונות. למעשה – משכפל.

#include>stdio.h <
FILE *popen(const char *command,const char *type); int pclose(FILE *fp);: פתיחה וסגירה של פייפ.
fread: קריאת נתונים מתוך קובץ למסך. fgets: קריאת נתונים מקובץ לbuffer.

Kill לתהליך מס' 1. ניתן לכתוב במקום האחוז את הפי אי די של התהליך הריגה-של קיל. שולח פסיקה- Kill –KILL %1 –
gcc filename – a.out קימפול ויצירת קובץ הרצה בשם
-o: renaming the exe file

- PTHREAD Library

#include<pthread.h>
int pthread_create(pthread_t*, pthread_attr_t*, void*(func_name),void* arg) -יצירת תהליך
הפרמטרים שפונ זו מקבלת: 1 – מצביע למקום בו יאוחסן ה thread-החדש. 2- מאיינים שמתארים את ה thread-החדש, בד"כ נותנים NULL לתהליך

חדש. 3- מצביע לפונ' אותה התהליך יבצע. 4 -רשימת ארגונטים שתסופק לפונ.'
פונ' לסיום thread , מקבלת סטטוס סיום ואין ערך מוחזר מהפונ'. סיום רק תהליך ששלח את הפקודה הזו– pthread_exit (status *void)
קבלת המזהה של ה-thread שלי, מחזיר pthread_t: pthread_self()
המתנה לתהליך אחר, מחזיר 0 במקרה והצליח, אחרת כישלון: pthread_join int (pthread_t ,void* status)
ה-thread מוותר על זמן CPU שלו ונותן אותו לאחרים pthread_yield() -

יצירת מנעול, אפשרות רגילה-רק אחד כותב בו זמנית – pthread_mutex_t mutexName=PTHREAD_MUTEX_INITIALIZER
נעילת מנעול, מקבל את כתובת הmutex- עליו עושים נעילה –pthread_mutex_lock(&mutexName)
שחרור מנעילה, המנעול משוחרר ותהליך אחר יכול להשתמש בו -pthread_mutex_unlock(&mutexName)
שיחרור מהזיכרון, אף אחד כבר לא יכול להשתמש בו - pthread_mutex_destroy(&mutexName)

- Shell Script

על מנת להגדיר shell משתמשים כותבים: (בד"כsh סוגה shell /bin!#
משתנים מסומנים ב-$. פרמטרים של השוואה: -lt, -gt, -eq, -le- ,ne, -ge
מדפיס למסך echo -
שרשור מחרוזת למשתנה- שגם הוא מחרוזת נעשה בלי מרכאות.
התנאי if נסגר ע"י fi. ניתן להשתמש גם ב:fi if else.
פעולה מתמטית מתבצעת ע" expr.
משתנים מיוחדים: #$-מספר ארגומנטים בשורת הפקודה, $0 שם התוכנית, $1-$9 - ארגומנטים בשורת הפקודה, @$-כל הארגומנטים (מופרד במרכאות),
*$-כל הארגומנטים, $$-מספר ה-id של ה-script שרץ.

- Unix Command Shell

echo	Write arguments to STDOUT
time at date	Time how long a command takes to execute Execute commands at a specified time Display date and time
uname clear	Display system information Clear the screen
script	Make a record of a terminal session
du df	Summarize disk usage for given files/directories Display information about free and used disk space
uptime calendar cat	Show how long the system has been up Reminder service Write files to STDOUT
more less	Paginate a file to STDOUT Similar to more. More features
head	Display first n lines
tail lpr	Display last n lines of a file Print a file
sort spell wc	Sort files. See option -u Spell-check a file Count bytes, lines, and words in a file
ls cd	List directory contents. See options –a, -l and -F Change directory. Accepts absolute and relative path names.
mkdir rmdir	Make a new directory Remove a directory
cp mv rm find	Copy a file/directory. See option -i Move (rename) a file/directory to another name/location . See Option -r Removes files. See options –i and -r Find files

pwd	Print the working (current) directory
ln	Make a link to a file/directory. See option -s
umask who	Get or set file mode creation mask. More at chmod Display users currently logged on
finger whoami	Display information about users Display effective current user name
telnet ssh	Open a telnet connection Open a SSH connection
ftp logout/exit/ctrl-d	Open an FTP connection End a session

- Files, Macro, Makefile
  - Files: filedescriptors מטיפוס FILE*:

STDIN	0	קליטת נתונים מהמקלדת
STDOUT	1	הדפסת נתונים על המסך
STDERR	2	הדפסת נתונים על המסך

EOF = -1 נמצאים בסוף הקובץ

FILE* fopen(char* filename, char* mode) : פותח קובץ ומחזיר את ה של הקובץ או אם לא הצליח
int feof(FILE* stream) : מזהה אם קובץ הסתיים אם כן מחזיר 0 אחרת מספר אחר
int fflush(FILE* stream): -כתיבה בזמן אמת לתוך קובץ מחזיר 0 אם הצליח ואחרת 1
int fclose(FILE* stream): -סגירת קובץ מחזיר 0 אם הצליח ואחרת 1
int fscanf(FILE* stream,char* format,other arguments): -קליטת נתונים מקובץ מחזיר 0 אם הצליח ואחרת 1
int fgetc(FILE* stream): -קליטת תו יחיד מחזיר 0 אם הצליח ואחרת 1
int fgetchar(): -קליטת תו מהמקלדת מחזיר 0 אם הצליח ואחרת 1
char* fgets(char* s, int n, FILE* stream): -קליטת מחרוזת מחזיר 0 אם הצליח ואחרת 1
int fprintf(FILE* stream, char* format, arguments): -הדפסה לתוך קובץ מחזיר 0 אם הצליח ואחרת 1
int fputc(int c, FILE* stream): -הדפסת תו לתוך קובץ מחזיר 0 אם הצליח ואחרת 1
הדפסת מחרוזת - int putchar() : -הדפסת תו על המסך מחזיר 0 אם הצליח ואחרת 1
char* fputs(char* s,FILE* stream)
int fseek(FILE* stream, long offset,int wherefrom): ממקם את מקום ההצבעה בקובץ ומחזיר אותו
-קבועים לשימוש במקום ה wherefrom
מהתחלה:SEEK_SET
הנוכחי מהמקום:SEEK_CUR
מהסוף:SEEK_END
מחזיר את המיקום בקובץ
מחזיר מצביע להתחלה - long ftell(FILE* stream)
void rewind(FILE* stream)
-קריאה מתוך קובץ מחזיר 0 אם הצליח ואחרת 1
int fread(void* ptr, size ,FILE* stream)
-כתיבה לתוך קובץ מחזיר 0 אם הצליח ואחרת 1
מחיקת קובץ - int fwrite(void* ptr, size ,FILE* stream)
שינוי שם קובץ - int remove(char* filename)
int rename(char* oldname,char* newname)

- Macro

#include פניה לספריה
#define הגדרת משתנה
#undef מחיקת משתנה
#if תנאי
#ifdef אם מוגדר
#ifndef אם לא מוגדר

- Makefile

make מנגנון המבצע יצירה של קובץ הרצה