

| Línea de Comandos | |
|--|--|
| Comandos | cat, wc, head, tail, tr, sort, cd, ls, mkdir, rmdir, cp, find, pwd |
| Secuencias de comandos | <ul style="list-style-type: none"> Ejecuta el segundo comando solo si el primero falla. && Ejecuta el segundo comando solo si el primero tiene éxito. ; () Ejecuta comandos en una sub-shell. |
| Redirecciones y tuberías | <ul style="list-style-type: none"> Envía la salida de un comando como entrada del siguiente. > Redirige la salida estándar de un comando >> Redirige la salida estándar de un comando (append) 2> y 2>> Redirección de la salida estándar de error |
| Expresiones regulares (grep) | |
| Caracteres y grupos | a, b, [aA], [0-9], [A-Za-z], [:blank:], [:alnum:] |
| Posicionamiento (anclas) | <ul style="list-style-type: none"> ^ principio de línea \$ final de línea \< principio de palabra \> final de palabra |
| wildcards | <ul style="list-style-type: none"> . cualquier carácter * el patrón anterior 0 o más veces + el patrón anterior 1 o más veces |
| Repeticiones | {N} {N,} {N,M} el patrón se repite N veces, N veces o más, N veces y no más de M |
| Programación en bash | |
| Argumentos | <ul style="list-style-type: none"> \$1, \$2, ... argumentos posicionales \$# número de argumentos \$@ todos los argumentos |
| Condicionales | <pre>if [[condición]]; then comandos elif [[otra_condición]]; then comandos else comandos fi</pre> |
| Bucles | <pre>for variable in lista; do comandos done</pre> |
| Operaciones con cadenas | |
| | atoi(3), sprintf(3), strlen(3), strcat(3), strcpy(3), strcmp(3) |
| Gestión de errores | |
| perror(3) | Imprimir en stderr el mensaje asociado a valor de la variable errno fijado por la última llamada |
| strerror(3) | Obtener el mensaje asociado a un código de error |
| Información del sistema, usuario | |
| uname(2) | Información del sistema (SO, host, versión del kernel...) |
| getuid(2), geteuid(2) getgid(2), getegid(2) | Obtener identificador real o efectivo del usuario/grupo del proceso |
| getpwnam(3), getpwuid(3) | Obtener entrada del archivo passwd del usuario |

| Sistema de Fichero - Comandos | |
|--------------------------------------|--|
| stat(1) | muestra información sobre el fichero o sistema de ficheros |
| touch(1) | actualiza los tiempos de acceso y modificación |
| chown(1) | cambia el propietario (usuario y/o grupo) |
| chmod(1) | cambia el modo (permisos) |
| ln(1) | creación de enlaces simbólicos y duros |
| readlink(1) | leer el path de un enlace simbólico |

| Sistema de Fichero - Llamadas al Sistema | |
|---|--|
| open(2) | Abrir un fichero |
| close(2) | Cerrar un descriptor abierto |
| write(2), read(2) | Escribir o leer de un descriptor previamente abierto |
| lseek(2) | Posicionar un fichero |
| stat(2), lstat(2), fstat(2) | Obtener el estado de un fichero (ver inode(7) para macros y funciones extra) |
| link(2), symlink(2) | Crear enlaces rígidos o simbólicos |
| unlink(2) | Eliminar un nombre de fichero y posiblemente el fichero al que se refiere |
| opendir(3) | Abrir un directorio |
| readdir(3) | Leer entradas de un directorio |
| closedir(3) | Cerrar un directorio |
| fsync(2) | Sincroniza un fichero en el almacenamiento secundario |

| Procesos - Comandos | |
|----------------------------|--|
| ps(1) | muestra información sobre los procesos |
| kill(1) | Envía una señal a un proceso o conjunto de procesos |
| chrt(1) | cambia/consulta la política y prioridad de planificación de un proceso |
| nice(1) | Ajusta el valor de <i>nice</i> de un proceso |
| jobs(1p), fg(1p), bg(1p) | Control de trabajos en la shell & Ejecuta el comando en segundo plano (background). |

| Procesos - Llamadas al Sistema | |
|---------------------------------------|---|
| getpid(2), getppid(2) | Obtener el identificador del proceso o del proceso padre |
| getpgid(2), setpgid(2) | Obtener o establecer el identificador del grupo de procesos de un proceso |
| getsid(2) | Obtener el identificador de sesión del proceso |
| setsid(2) | Crear una sesión y establecer el id. del grupo de procesos del proceso |
| fork(2) | Crear un proceso hijo pid_t pid = fork(); switch (pid) { case -1: perror("fork"); exit(1); case 0: // Proceso Hijo |

| | |
|-----------------------------------|--|
| | <pre> break; default: // Proceso Padre break; } </pre> |
| _exit(2) | Terminar un proceso |
| wait(2), waitpid(2) | Esperar a que termine un proceso hijo y obtener su estado |
| execve(2), execlp(3)... | Ejecutar un programa |
| getcwd(3), chdir(2) | Consultar y cambiar el directorio de trabajo |
| getenv(3), setenv(3), unsetenv(3) | Obtener, establecer o eliminar variables de entorno |
| kill(2) | <p>Enviar una señal. Señales importantes:</p> <ul style="list-style-type: none"> • SIGINT: Interrupción. Se puede generar con Ctrl+C • SIGSTOP: Parar proceso. • SIGTSTP: Parar proceso. Se puede generar con Ctrl+Z • SIGCONT: Reanudar proceso parado • SIGKILL: Terminación brusca. • SIGSEGV: Violación de segmento de datos • SIGTERM: Terminar proceso • SIGCHLD: Terminación del proceso hijo |

| Memoria | |
|-----------------------------------|--|
| mmap(2) munmap(2) mremap(2) | Crea/destruye/redimensiona un segmento de diferentes tipos. |
| brk(2) | Mueve la localización del <i>program break</i> , aumentando/disminuyendo el <i>heap</i> del proceso. |
| msync(2) | Sincroniza un fichero con un mapa de memoria |
| pmap(1) | Muestra el mapa de memoria de un proceso, también en /proc/<pid>/maps |
| malloc(3) | Asigna memoria dinámica |
| free(3) | Libera memoria dinámica |

| POSIX Threads | |
|--|--|
| pthread_create(3) | Crea un nuevo thread |
| pthread_join(3) | Sincroniza la finalización de un thread |
| Mutex | |
| pthread_mutex_init(3) | <p>Inicializa un mutex. Inicialización estática:</p> <pre>pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;</pre> |
| pthread_mutex_destroy(3) | Destruye un mutex |
| pthread_mutex_lock(3) pthread_mutex_unlock(3) pthread_mutex_trylock(3) | Funciones para el bloqueo y desbloqueo de mutex |
| Mutex de Lectura/Escritura | |
| pthread_rwlock_init(3) | <p>Inicializa un mutex de lectura/escritura. Inicialización estática:</p> <pre>pthread_rwlock_t rwlock = PTHREAD_RWLOCK_INITIALIZER;</pre> |
| pthread_rwlock_destroy(3) | Destruye un mutex de lectura/escritura |
| pthread_rwlock_[rd wr]lock(3) | Funciones para el bloqueo y desbloqueo del mutex para operaciones de |

| | |
|--|---|
| <code>pthread_rwlock_unlock(3)</code> | lectura (rd) o escritura (wr) |
| Variables de Condición | |
| <code>pthread_cond_init(3)</code> | Inicializa una variable de condición Inicialización estática: <code>pthread_cond_t cond = PTHREAD_COND_INITIALIZER;</code> |
| <code>pthread_cond_destroy(3)</code> | Destruye una variable de condición |
| <code>pthread_cond_wait(3)</code> | Espera en la variable de condición a que se cumpla un predicado <pre>pthread_mutex_lock(...) while (<predicado es falso>) { pthread_cond_wait(...) } ... pthread_mutex_unlock(...)</pre> |
| <code>pthread_cond_signal(3)</code> | Despierta un thread esperando en una variable de condición |
| <code>pthread_cond_broadcast(3)</code> | Despierta todos los threads esperando en una variable de condición |