#	Adattípus	Létrehozás	Írás / Küldés	Várakozás	Olvasás / Fogadás	Bezárás
File	f FILE*	f = fopen (char*, int mode)	fwrite (&c, sizeof(c), sizeof(c), f)	-	fread (&c, sizeof(c), sizeof(c), f)	fclose(f)
Fork	pid_t pid (=int)	pid = fork() pid = 0 : child pid > 0 : parent	*visszatérö hivas* system(char* exec) *nem tér vissza* execv(char* exec, char** args)	waitpid (child, &status, 0)	*parent* getppid() *saját* getpid()	*saját magának* raise(signal)
Signal	struct sigaction	sigemptyset(&sigset)	kill(pid, signal)	pause() sigsuspend(&sigset)	sigaction.handler = handler(signal)	
	struct sigset	sigfillset(&sigset)			sigaction(signal, &sigaction, NULL)	
		sigdelset(&sigset, signal)			sigprocmask(SIG_UNBLOCK, &sigset, NULL)	
Signal Data	union sigval sv_ptr;	sv_ptr.sival_ptr = &message	sigqueue (getppid(), SIGTERM, sv_ptr)		sigaction.handler = void handler(int signumber, siginfo_t *info, void *nonused)	
Timers	struct itimerval t	t.it_interval.tv_sec = 3 t.it_value.tv_sec = 1	setitimer (ITIMER_REAL, &t, NULL)	getitimer (ITIMER_REAL, &timer)	getitimer (ITIMER_REAL, &timer)	it_interval.tv_sec = 0
Pipe	int pipefd[2]	pipe(pipefd)	write (pipefd[STDOUT_FILENO], &msg, size)	*lásd: poll*	read (pipefd[STDIN_FILENO], msg, size)	close (pipefd[STDOUT_FILENO])
Named Pipe	int fifo	fifo = mkfifo (char* name, S_IRUSR | S_IWUSR)	fd = open (fifo, O_WRONLY)		fd = open(fifo, O_RDONLY)	close(fd)
			write (fd, msg, sizeof(msg))		read(fd, msg, sizeof(msg))	unlink(fifo)
Poll	struct poll_fds[]	poll_fds[0].fd = file poll_fds[0].events = POLLIN | POLLOUT	result = poll (poll_fds, 1, timeout) *result = 0 : timeout* *result > 1 : POLLIN*	-	if (poll_fds[0].revents & POLLIN) { // read }	*timeout*
S-V MQ	key_t key = ftok (char* , 1)	int mq = msgget (key, 0600 | IPC_CREAT)	msgsnd (mq, &msg, msg_size , 0)	-	msgrcv (mq, &msg, msg_size, mtype, 0)	msgctl (mq, IPC_RMID, NULL)
	struct msg = { long mytpe; ... } msg_size = sizeof(struct msg) - sizeof(long)					
essage Queue	struct mq_attr mq_name = "/queue"	mqd_t mq_des = mq_open (mq_name, O_CREAT | O_RDWR, 0600, &attr)	mq_send (mq_des, msg , sizeof(msg), prioirty)	-	mq_receive (mq_des , target_buffer, sizeof(msg), 0)	mq_close(mq_des) mq_unlink(mq_name)

Tatai Áron, 2023